ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DOMÓTICO BASADO EN SOFTWARE LIBRE PARA UNA VIVIENDA UNIFAMILIAR

(Development and implementation of a domotic system based in open source software for a single family home)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Iván Sarmiento Montenegro

Septiembre - 2019

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Iván Sarmiento Montenegro
Director del TFG: José María Zamanillo Sainz de la Maza
Título: "Desarrollo e implementación de un sistema domótico basado en software libre para una vivienda unifamiliar"
Title: "Development and implementation of a domotic system based in open source software for a single family home"

Presentado a examen el día: 30 de septiembre de 2019

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente: Zamanillo Sainz de la Maza, José María
Secretario: Pereda Fernández, José Antonio
Vocal: Fernández Ibáñez, Tomás

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

“A mi familia, incluyendo a los que nos faltan: sin todos vosotros esto no habría sido posible. Y en especial: a mi padre, por tu inspiradora insistencia; a Zama, por tu inestimable ayuda; a María, que eres una extensión de mi persona; y a Lucía, que todavía no has llegado y ya eres el motor de nuestras vidas.”

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN

La automatización del hogar, también conocida como domótica –proveniente del latín "*domus*" (casa) y *automática*- está definida en el diccionario de la RAE como el "*conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda*"[1]. Se trata por tanto de un ámbito donde la ingeniería tiene un recorrido muy amplio que se lleva desarrollando en las últimas décadas, si bien en los últimos 5-10 años ha vivido un crecimiento considerable gracias a la adopción de distintos estándares, la presencia cada vez más generalizada de redes de comunicaciones sin cables en los hogares y el abaratamiento de la mayoría de los dispositivos involucrados: sensores, actuadores, controladores, etc.

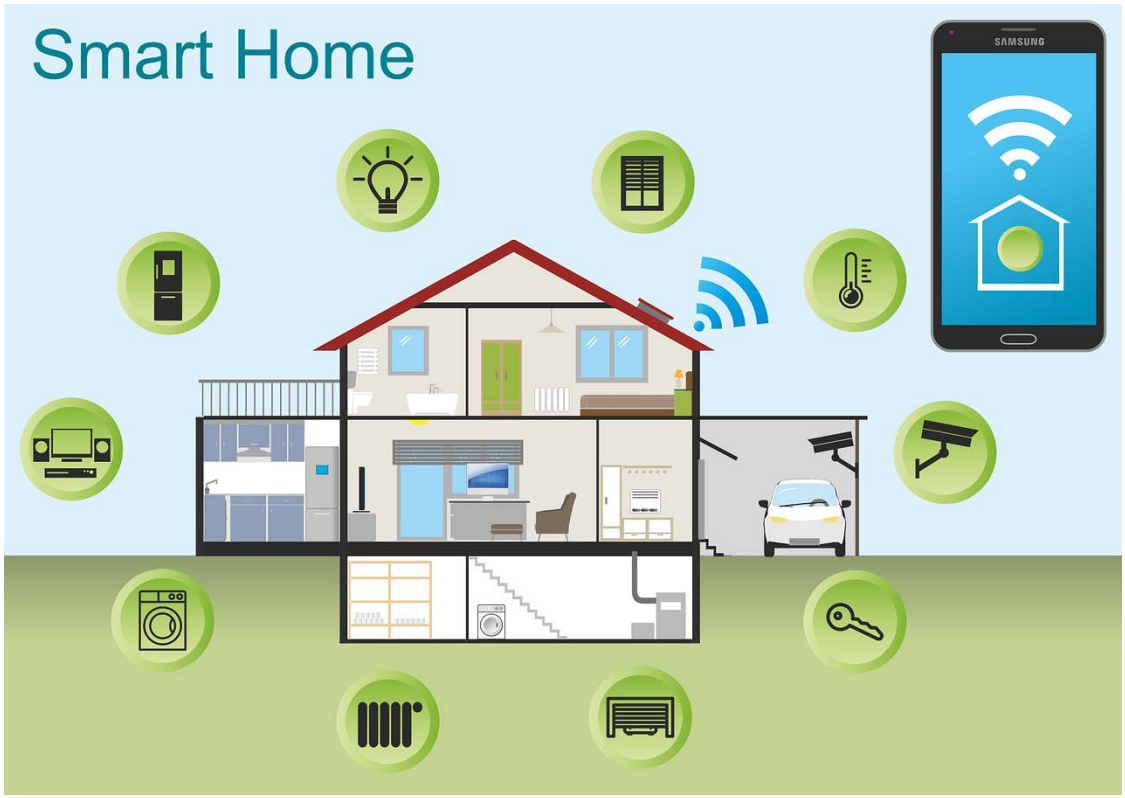

Figura 1. Representación de algunos de los sistemas que pueden ser automatizables en un hogar actual.

Algunos autores afirman que "*la domótica es una de las formas en que la automatización busca mejorar el estado actual de la sociedad aumentando la calidad de vida de la humanidad.*" [1]

Los aspectos en los que este nuevo ámbito de trabajo de la ingeniería en las viviendas domotizadas o inteligentes ("*Smart Home*") puede mejorar la calidad de vida de las personas se podrían resumir en:

- ***Ahorro de recursos***: los sistemas se encienden y apagan de forma automática cuando es necesario basándose en programaciones predefinidas por los usuarios, eventos desencadenantes o incluso el "aprendizaje" de los hábitos y rutinas diarias. Esto se aplica a la iluminación, pero también a la gestión ambiental del hogar mediante termostatos inteligentes o a los sistemas de riego conectados.

[1] https://dle.rae.es/srv/fetch?id=E7W0v9b

- ***Comodidad y confort***: la automatización de tareas rutinarias (como la apertura y cierre de las persianas o el encendido y apagado manual de electrodomésticos en modo reposo) posibilita que queden fuera de las actividades de los residentes, evitando las posibles molestias asociadas y liberando tiempo para otros menesteres. En cierto modo podría hacerse el paralelismo con la aparición de los mandos a distancia para los receptores de TV.
- ***Seguridad y tranquilidad***: además de las aplicaciones específicas en el ámbito de la seguridad y vigilancia, también se garantiza que las tareas mencionadas anteriormente se ejecuten de manera adecuada, evitándose por tanto descuidos y olvidos que podrían tener consecuencias negativas para el hogar: un jardín encharcado, una luz exterior encendida durante una o varias noches, una ventana o puerta abierta al salir de casa…
- ***Información y control sobre lo que ocurre en el hogar***, tanto a nivel de seguridad como ambiental, consumos energéticos, etc. Los datos permiten tomar decisiones para mejorar la habitabilidad y/o reducir el gasto, como se ha comentado inicialmente.

La gran variedad de sistemas, componentes e incluso tecnologías involucradas en la automatización de una vivienda suele ser uno de los puntos críticos a la hora de afrontar un proyecto amplio de domotización de la misma. Generalmente, se trata de plataformas comerciales que suelen resultar bastante caras, tanto sus controladores centrales (también conocidos como "centralitas", "concentradores" o "*hubs*") como el resto de dispositivos. Además son bastante restrictivas en cuanto a la personalización o la mejora de sus funcionalidades.

Otro problema es que habitualmente se centran en uno de los múltiples ámbitos del hogar (la iluminación, la seguridad, las persianas, el control ambiental, etc.), por lo que para conseguir un sistema completo resulta necesario tener varias soluciones trabajando en paralelo, con el consiguiente incremento en el coste global y la multiplicación de las interfaces de control para cada uno de ellos.

El objetivo principal del proyecto desarrollado es, precisamente, combatir esa debilidad: **unificar toda la información y funcionalidad distribuida en el hogar bajo un único sistema**. De este modo tanto el control como la visualización de información por parte de los usuarios se haga desde un único punto.

La vivienda en la que se ha instalado el sistema domótico caracterizado en este Trabajo Fin de Grado es un chalet de dos plantas con unos 140 m^2 útiles, construido en el año 2014. Está ubicada en Santander, con orientación al norte, este y sur. Posee calificación energética B gracias a, entre otras cosas, su sistema de calefacción de suelo radiante mediante bomba de calor.

Este documento pretende dar una visión general de la filosofía y sistemas empleados, sin profundizar demasiado en cada apartado. El motivo es que muchas de las áreas a tratar prácticamente requerirían un documento monográfico igual de extenso que el presente para ser cubiertas de manera más específica, quedando fuera del objetivo principal del presente trabajo.

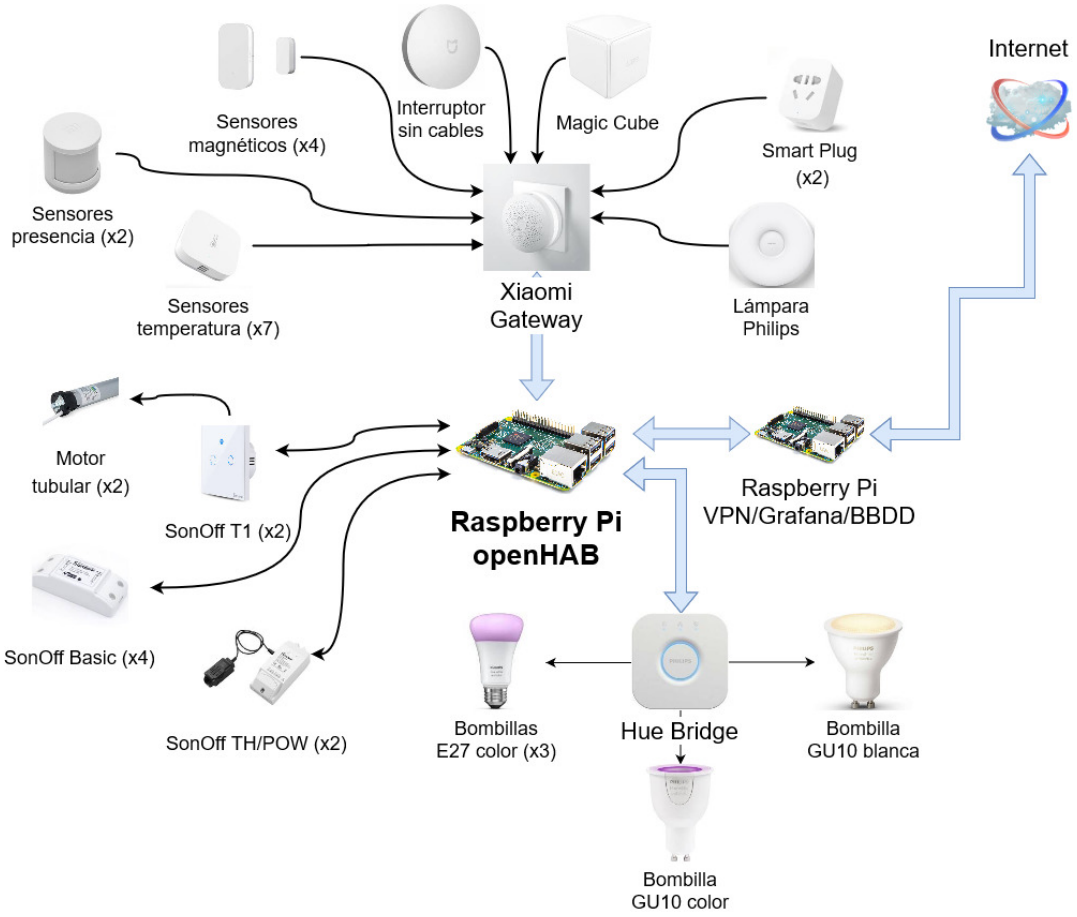

Figura 2. Esquema conceptual[2] de la automatización en la vivienda: dispositivos conectados.

Nota: todas las imágenes y figuras del presente trabajo en las que la fuente no esté citada expresamente son obra del autor, o tienen licencias que permiten su reutilización sin citar el origen (libres de derechos).

[2] Realizado con la herramienta draw.io (https://www.draw.io/)

2. ALTERNATIVAS PARA LA CENTRALIZACIÓN DE SISTEMAS DOMÓTICOS

Existen diversas plataformas que permiten alcanzar el objetivo principal mencionado en el apartado anterior. Todas ellas presentan similares funcionalidades, filosofías de desarrollo y complejidad a nivel técnico, por lo que en un principio puede resultar complicado elegir qué sistema se convertirá en el centro de la instalación.

Las tres que parecen tener mayor penetración actualmente entre las soluciones de código abierto, que se consideró una condición esencial para garantizar la posibilidad de personalización del código y la continuidad de la solución incluso si todos los desarrolladores abandonaran su adopción. Todas se sostienen gracias a comunidades de usuarios numerosas y activas, además de las aportaciones privadas, y son las que se han valorado y probado antes de tomar una decisión definitiva.

- **Domoticz**[3]

Escrita en C++ y diseñada para poder ejecutarse sobre distintos sistemas operativos, incluyendo Linux, Windows, Mac OS y otros sistemas embebidos como Raspberry Pi, equipos de almacenamiento en red (NAS). La interacción con el usuario se realiza mediante HTML5 a través de un servidor web integrado, y el diseño general está enfocado a la sencillez para la adopción.

- **HomeAssistant**[4]

Desarrollada en Python, compatible también con prácticamente cualquier tecnología de servidor para la instalación local. Las prioridades del equipo de desarrollo son el control de la información a nivel local, huyendo del almacenaje de datos en la nube, la privacidad y la seguridad.

- **openHAB**[5]

Programada en Java, se caracteriza por la gran cantidad de tecnologías y marcas soportadas gracias a su arquitectura basada en "*plugins*": pequeños módulos de código desarrollados para comunicarse con familias o dispositivos específicos. Comparte con las anteriores la compatibilidad para ser instalada en diversas plataformas, así como la flexibilidad para su control por parte de los usuarios.

Tras una fase de valoración de las alternativas comentadas, finalmente se tomó la decisión de utilizar openHAB por presentar algunas ventajas relacionadas con la experiencia previa del autor, tanto académica como profesional, y con las particularidades del proyecto en mente para la vivienda en cuestión:

- Aprovechar los conocimientos de Java del autor, gracias a lo aprendido en diversas asignaturas de la titulación que, pese a haber sido una introducción al lenguaje y sus

[3] https://www.domoticz.com/

[4] https://www.home-assistant.io/

[5] https://www.openhab.org/

particularidades, resultan suficientes para tener una familiaridad con el mismo que no se da en las otras opciones.[6]

- La **compatibilidad garantizada** con algunos de los dispositivos que ya estaban instalados en la casa, como la iluminación basada en el sistema HUE de Philips o los relés inteligentes SonOff del fabricante Itead.
- La garantía de mantener un **funcionamiento estable** corriendo sobre sistemas basados en Raspberry Pi, con los que el autor también está ampliamente familiarizado tras años de experiencia en diversas aplicaciones, llegando incluso a impartir charlas de divulgación y formación no reglada en el ámbito de los Cursos de Verano de la Universidad de Cantabria en numerosas ediciones.[7]

[6] Por ej., Sistemas Informáticos: https://web.unican.es/estudios/Documents/Guias/2018/es/G818.pdf

[7] https://web.unican.es/cursosdeveranoyextension/cursos-de-verano/curso?c=2773

3. DESCRIPCIÓN DEL SOFTWARE OPENHAB

La "**open H**ome **A**utomation **B**us" (aproximadamente, "*Canal de intercambio de información Abierto para la Automatización Doméstica*") es una plataforma de automatización doméstica de código abierto agnóstica a la tecnología. Es decir, que pretende integrar multitud de dispositivos y sistemas domóticos independientemente de la tecnología en la que estén desarrollados o su marca de fabricante, aunando toda su gestión bajo un único sistema.

Comenzó su andadura en 2010 de la mano de Kai Kreuzer[8] , un "*Java Software Architect*" y profesional en el ámbito del IoT[9]. Está desarrollada en lenguaje Java. Cuenta con una importante comunidad de usuarios, desarrolladores y patrocinadores a través de la Fundación openHAB[10], incluidas numerosas empresas del ámbito de la automatización y el IoT.

La plataforma ha centrado siempre su desarrollo en su habilidad de integrar multitud de tecnologías, sistemas y dispositivos bajo una solución única, proporcionando además una interfaz de usuario y una aproximación a las reglas de automatización comunes para cualquiera de ellos. Y todo ello tratando de mantener una gran flexibilidad para poder adaptarse a las necesidades de cada usuario, permitiéndole alcanzar las soluciones necesarias para cualquier caso concreto.

Sin embargo estas características tan deseables tienen asociado un coste: la necesidad, por parte del usuario que desee instalar la solución en su hogar, de disponer de unos conocimientos técnicos básicos, ganas de "cacharrear" y tiempo que poder dedicarle. La curva de aprendizaje, aunque asequible, puede resultar bastante pronunciada en los comienzos, resultando el apoyo de la comunidad de usuarios a través de las numerosas guías y tutoriales disponibles, o de los foros, una ayuda inestimable, tanto para noveles como para veteranos.

Los bloques principales que sustentan cualquier instalación de openHAB son:

- **Cosas ("*Things*")**: son entidades que se pueden añadir físicamente al sistema, y que pueden incorporar más de una función específica (por ejemplo, un sensor de temperatura que también proporciona valores de humedad y presión atmosférica). No tienen por qué ser necesariamente un dispositivo físico, aunque es lo más habitual, porque también puede ser un servicio web o cualquier otra fuente controlable de información y/o funcionalidad.
- **Uniones ("*Bindings*")**: entidades lógicas que se pueden considerar como adaptadores software, haciendo que las Cosas estén disponibles para el sistema de automatización. Son módulos independientes que vinculan las Cosas con los Objetos, proporcionando una capa de abstracción lógica que elimina la necesidad de considerar para cada caso los requisitos a nivel de comunicaciones de modo que éstas pueden ser tratadas de forma genérica por el entorno de trabajo.
- **Canales ("*Channels*")**: donde las Cosas publican sus funcionalidades, y que están definidos para cada una de ellas en su Unión. No es preciso definir todos los canales que estén incluidos en la Unión de una Cosa, pero sí que será necesario hacerlo si se quiere hacer uso de alguna funcionalidad.

---

[8] http://www.kaikreuzer.de/

[9] Internet of Things, Internet de las Cosas.

[10] https://www.openhabfoundation.org/

- **Ítems (“*Items*”)**: representan las funcionalidades que pueden ser utilizadas por el sistema, tanto a través de una interfaz de usuario como de la lógica de automatización. Los Ítems tienen asociado un Estado (“*State*”) y pueden recibir comandos.
- **Enlaces (“*Links*”)**: son el pegamento que une a los Ítems con las Cosas. Un Enlace es una asociación entre un único Canal y un Ítem. Si un Canal está vinculado a un Ítem se considera que está habilitado, por lo que la funcionalidad que ese Ítem representa será accesible a través de ese Canal.
- **Mapas de Sitio (“*Sitemaps*”)**: son colecciones de Ítems organizadas por el usuario a su conveniencia, de forma que agrupan la funcionalidad para ser usada en distintas interfaces de usuario.
- **Persistencia (“*Persistence*”)**: es la posibilidad de guardar y mantener información en el tiempo, de modo que pueda ser consultada a posteriori (por ejemplo, para restaurar valores tras un reinicio del sistema, para calcular estadísticas o para hacer representaciones gráficas de la información. La información se guarda en bases de datos, y están soportadas las más populares. Es posible tener varias instancias de Persistencia funcionando de forma simultánea
- **Reglas (“*Rules*”)**: se utilizan para la automatización de procesos. Cada Regla puede ser desencadenada por algún evento en nuestro ecosistema, de modo que cuando se produzca se realice alguna tarea (por ejemplo, activar un interruptor, realizar un cálculo matemático y mostrarlo a través de una variable, activar otro conjunto de Reglas).

Se indican también las nomenclaturas en inglés porque es la que se utiliza de manera más generalizada en la documentación, incluso si ésta es en otros idiomas.

Esta estructura puede parecer complicada, al menos inicialmente, pero en cuanto se empieza a trabajar con los distintos conceptos implementando las distintas entidades resulta bastante dinámica y sencilla de mantener. La siguiente imagen puede ayudar a comprender cómo funciona.

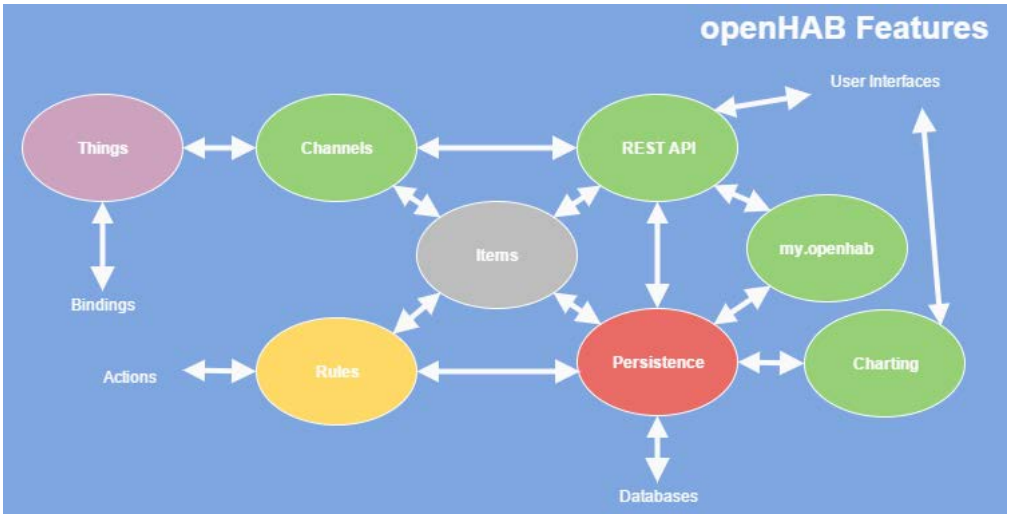

Figura 3. Esquema de funcionamiento de los distintos bloques de openHAB.[11]

[11] Fuente: https://community.openhab.org/t/lets-talk-about-oh-2-drawings/13096

4. CONTROLADOR CENTRAL: OPENHABIAN INSTALADO EN RASPBERRY PI

Tras definir de forma teórica el concepto del sistema y de hacer una breve introducción a la solución elegida, a partir de este apartado el trabajo se centra en describir las distintas partes de la instalación realizada.

Figura 4. Raspberry Pi 2 modelo B: el corazón de la instalación

El primer paso es elegir una plataforma para alojar el software openHAB, que será el controlador central de la instalación. Como se ha visto en la descripción de sus características, hay una gran variedad de alternativas. Se ha elegido utilizar la plataforma Raspberry Pi 2 Model B, un equipo SoC[12] porque cumple los siguientes requisitos:

- **Consumo muy bajo**. Teniendo en cuenta que el sistema va a estar funcionando 24 horas al día, 7 días a la semana, 365 (o 366) días al año, un consumo contenido se traduce en un menor gasto eléctrico. El consumo de una placa de este modelo, cuando no está en reposo, fluctúa entre los 2,6W y los 3,7W [2]. Es decir, en el peor de los escenarios, suponiendo un consumo de 4W, estaríamos ante un consumo de unos 35kWh al año. Considerando un precio conservador de la electricidad de 0,14€/kW./h., supondría menos de 5€ al año.
- **Pequeño tamaño**. Las dimensiones reducidas del equipo de 83x58x20mm, son una auténtica ventaja a la hora de poder ubicarlo prácticamente en cualquier lugar. En este caso está alojado en uno de las cajas de comunicaciones de la vivienda, junto con otros dispositivos de comunicaciones: la ONT[13] de la conexión de fibra óptica y un *switch* de red LAN que conecta con las tomas repartidas por las habitaciones, que también da servicio a la Raspberry.

[12]System on Chip, con todo el sistema en un único circuito.

[13]Optical Network Terminal.

Figura 5. Aspecto de la caja de comunicaciones que aloja el equipamiento de red del proveedor de Internet y la Raspberry Pi con la instalación de openHAB.

- **Estabilidad**. Se trata de un entorno con una estabilidad contrastada (incluso se utiliza en una iniciativa educativa en el espacio, a bordo de la Estación Espacial Internacional: la Astro Pi[14]). En los más de dos años que lleva funcionando de manera ininterrumpida en el caso que nos ocupa no ha sufrido ningún fallo de funcionamiento: ni cuelgues, ni reseteos aleatorios, ni cualquiera de los problemas habituales que suelen afectar a los equipos informáticos. Y además con un mantenimiento mínimo, puesto que las actualizaciones no son realmente necesarias como comentaremos más adelante en el apartado de seguridad.
- **Potencia de computación** suficiente. Pese a tratarse de un equipo pequeño y de bajo consumo, posee un hardware que le permite realizar cualquiera de las tareas que se espera pueda desempeñar un equipo de escritorio estándar: un procesador ARM de cuatro núcleos a 900 MHz. y 1 Gb. de RAM. En este caso los requisitos son todavía menores dado que el sistema se utiliza en una configuración "*headless*": sin monitor ni interfaz gráfica, accediendo siempre de forma remota para las operaciones de mantenimiento, lo cual reduce los requisitos de rendimiento. No se ha dado ninguna circunstancia en la que se pudiera sospechar que el entorno presentaba escasez de recursos para cumplir su cometido adecuadamente.
- **Sencillez** de mantenimiento y alta disponibilidad. El bajo coste del sistema (unos 20-30€) y del almacenamiento en forma de tarjeta SD de alta velocidad y de 8-16 Gb. de capacidad (menos de 10€) hacen que tener redundancia para afrontar posibles fallos sea extremadamente asequible, en contrapunto a una centralita tradicional de domótica que difícilmente sería inferior a los 300€. Tener copias del sistema a nivel software es tan sencillo como tener una tarjeta adicional de la misma capacidad que la usada en explotación y clonarla mediante cualquiera de los programas disponibles. En este caso se ha usado la utilidad "*Win32 Disk Imager*"[15], de licencia pública GPL-2[16].

[14] https://astro-pi.org/

[15] https://sourceforge.net/projects/win32diskimager/

- **Muy extendida** entre la comunidad de usuarios de openHAB. Es una de las soluciones más populares para alojar el servidor central del entorno, por lo que existen numerosos tutoriales y guías que facilitan mucho la instalación, personalización y mantenimiento del sistema. Además, al tratarse de una comunidad activa, los foros son un lugar idóneo para resolver problemas concretos que puedan surgir[17]. Este es un apartado en el que otras placas SoC, a pesar de poseer especificaciones son más potentes, no pueden competir con los más de 25 millones de Raspberry Pi vendidas.
- **Familiaridad**. Como se ha comentado anteriormente, la familiaridad del autor con la placa y algunas de sus aplicaciones motivó que fuera el entorno elegido para el despliegue. Y es que, además, ya había disponibilidad de algunos de estos equipos para poder realizar pruebas, desembocando en un entorno de producción final.

En cualquier caso, no todo son ventajas, y se pueden comentar algunos puntos débiles:

- **Rendimiento pobre** del sistema, especialmente del almacenamiento. Pese a utilizar tarjetas SD con velocidades de lectura y escritura altas -clase 10, como mínimo-, el acceso al sistema de almacenamiento no está muy optimizado: la tasa máxima de transferencia del "bus" SD es de unos 22Mb/s. tanto en lectura como en escritura[18], y una tarjeta SD está pensada para acceder a archivos de gran tamaño de forma secuencial, y no está a datos pequeños (4K) que es lo habitual en un sistema operativo. Esto se traduce en tiempos de reinicio de la placa y de arranque del sistema openHAB francamente mejorables. Pero las paradas del sistema son poco frecuentes, reduciéndose las programadas a un par de veces al año, a lo sumo; un número similar a las imprevistas (como los cortes en el suministro eléctrico, por ejemplo).

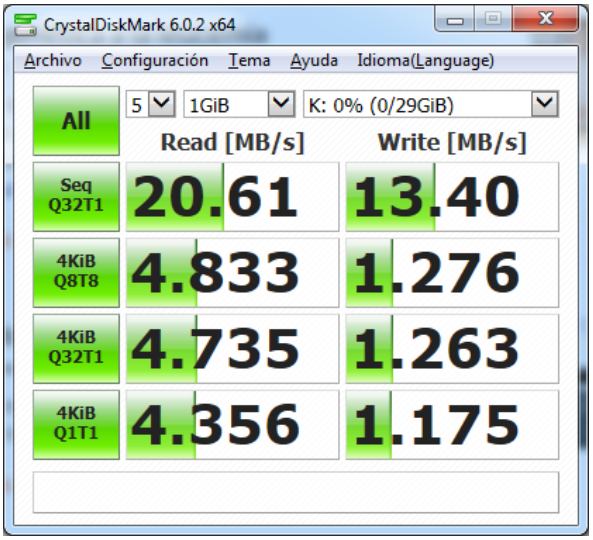

Figura 6. Test de rendimiento de una de las tarjetas SD utilizadas en la instalación, con la herramienta Crystal Disk Mark[19]*. Las velocidades para bloques de 4Kb son hasta 4 veces peores que en secuencial.*

---

[16] https://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.es.html

[17] https://community.openhab.org/

[18] https://elinux.org/RPi_SD_cards

[19] https://osdn.net/projects/crystaldiskmark/

- Posibilidad de **corrupción de datos** ante apagados inesperados. Otro de los puntos débiles de las tarjetas SD en las Raspberry Pi es que son susceptibles a sufrir corrupción de los datos almacenados si el sistema pierde el suministro eléctrico de forma inesperada en medio de una operación de escritura, debido a cómo se produce el proceso de acceso al disco. Una posible solución sería colocar una batería externa (con capacidad de alimentar lo que tenga conectado a la vez que se está cargando) entre la alimentación y el sistema, a modo de SAI, de modo que si se pierde el suministro eléctrico la Raspberry no pierda la alimentación al menos durante algunos minutos. En cualquier caso, es más que recomendable tener una tarjeta SD adicional con el sistema en producción clonado, además de realizar copias de seguridad periódicas de los archivos de configuración y demás información importante, paliándose así esta debilidad ante imprevistos.

4.1. PROCESO DE INSTALACIÓN HARDWARE

La instalación de la Raspberry Pi resulta extremadamente sencilla para utilizarla como servidor de openHAB.

Son necesarios los siguientes elementos, además de la propia placa:

- **Tarjeta micro SD**, que hará las veces de disco duro. Cuanto más rápida y fiable, mejor. El tamaño no resulta muy relevante, siendo suficientes 8 o 16Gb. (de hecho incluso es recomendable que no sea demasiado grande para que los procesos de copia de seguridad no sean muy largos ya que el método más fiable es realizar una imagen idéntica sobre otro soporte). La tarjeta alojará el sistema operativo, como se verá en el siguiente apartado referente a la preparación del software del equipo.
- **Alimentación**, en este caso un adaptador AC-DC de 5 V. y 1.2 A. para garantizar la estabilidad y suficiencia del suministro eléctrico del sistema.
- **Cable de red**. En la instalación del domicilio la red es Gigabit (1Gbps.), lo cual agiliza las comunicaciones entre los equipos, aunque no es un factor crítico en el caso que nos ocupa. Está conectado a un *switch* Ethernet, y éste a su vez al *router* que da servicio a la casa.

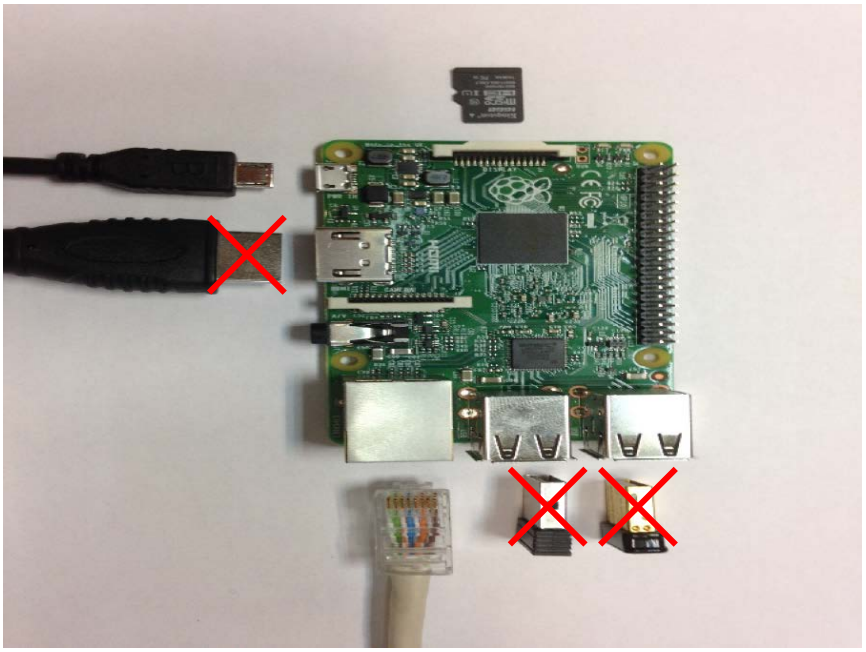

Figura 7. Raspberry Pi 2model B con sus accesorios más habituales; los que no son necesarios en la presente instalación están marcados: el teclado y ratón USB, un adaptador Wi-Fi y la conexión HDMI.

4.2. PROCESO DE INSTALACIÓN SOFTWARE

La instalación inicial del sistema operativo y del entorno openHAB es tremendamente sencilla gracias a la existencia de una distribución específica que aúna las dos cosas: **openHABian**. De este modo se elimina la necesidad de instalar un Linux en la tarjeta, configurarlo y después añadir la aplicación de domótica, junto con las configuraciones necesarias.

OpenHABian puede descargarse directamente desde la web del proyecto[20], copiarse en la tarjeta SD (por ejemplo utilizando el programa *Etcher*[21]) e introducirla en la Raspberry. Tras encenderla y esperar entre 15 y 45 minutos a que se instale y realice todas las operaciones de configuración predeterminadas, el sistema estará listo para usarse.

La conexión al mismo se hará a través de un navegador web utilizando el nombre de dominio (DNS[22]) que viene predeterminado: "openhab", en la dirección http://openhab:8080. También es posible establecer una conexión remota a través del protocolo SSH[23] (puerto 22), indicando el nombre DNS mencionado anteriormente o la dirección IP que se le haya asignado en nuestra red local, siendo las credenciales de acceso iniciales "openhabian//openhabian".

En la siguiente imagen se puede ver una captura de pantalla tras un inicio de sesión remota en el sistema utilizando el programa "*PuTTY*" como cliente SSH.

Figura 8. Pantalla inicial tras inicio de sesión, a través de SSH, en openHABian.

[20] https://www.openhab.org/docs/installation/openhabian.html

[21] https://www.balena.io/etcher/

[22] Domain Name Server.

[23] Secure Shell, protocolo criptográfico para las comunicaciones en redes informáticas.

En ningún momento es necesario conectar un periférico al equipo, resultando mucho más cómodo el acceso remoto desde otro equipo conectado a la red doméstica.

A partir de este punto ya es posible empezar a añadir y configurar los distintos elementos de nuestro ecosistema domótico, siendo posible hacerlo a través de dos vías: utilizando la interfaz de usuario llamada "*Paper UI*", que podría considerarse un asistente para la integración de elementos que ya estén conectados a la red domótica de alguna manera, y está accesible desde la dirección http://openhab/paperui/index.html; o editando directamente los archivos de texto de configuración, accesibles desde las carpetas compartidas por Samba (un protocolo de compartición de archivos en red) que vienen configuradas por defecto, explorando la red local y utilizando las mismas credenciales mencionadas anteriormente cuando nos las pida al abrir el sistema. Para el desarrollo del presente documento se utilizará esta segunda opción ya que, aunque puede resultar más complicada inicialmente, resulta más potente y personalizable a la larga.

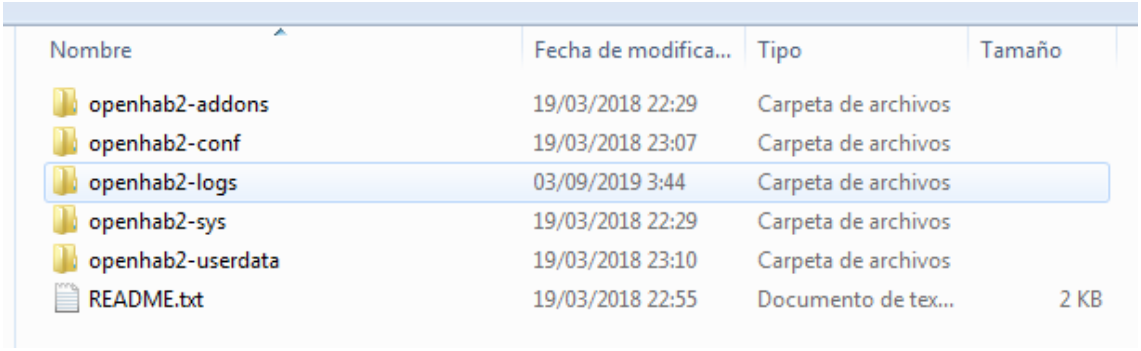

*Figura 9. Contenido del "*Samba share*" del sistema openHABian*

También es posible acceder al registro de eventos del sistema a través de un navegador web, lo cual puede resultar muy útil a la hora de verificar que no se están produciendo errores de funcionamiento y que los resultados obtenidos son los esperados. Este visor de eventos, conocido como "*frontail*", está disponible en la dirección local http://openhab:9001.

Tras la primera instalación es recomendable actualizar los paquetes del sistema operativo para aplicar todos los posibles parches de seguridad y rendimiento que estén disponibles, ejecutando primero el comando "*sudo apt-get update*" para actualizar la lista de paquetes del sistema, y "*sudo apt-get upgrade*" para instalar los nuevos paquetes encontrados[24]. Este proceso puede tardar bastante, pero dado que no se va a realizar muy a menudo resulta conveniente realizarlo en este primer momento.

Otra herramienta interesante que viene preinstalada por defecto en el sistema operativo openHABian es la de configuración: la "*openHABian Configuration Tool*". Se puede invocar mediante el comando "*sudo openhabian-config*", y dentro encontraremos numerosas opciones para configurar el controlador central de nuestro entorno domótico.

[24] http://manpages.ubuntu.com/manpages/trusty/es/man8/apt-get.8.html

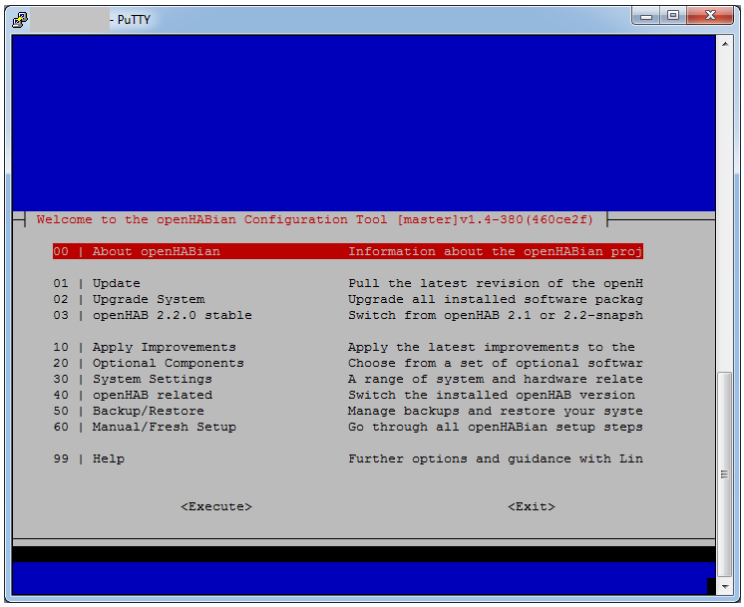

Figura 10. Pantalla principal de la herramienta de configuración de openHABian.

La incorporación de nueva funcionalidad al sistema mediante módulos añadidos ("*Add-ons*") se puede hacer de diversas maneras, pero la más sencilla es a través de la interfaz gráfica "Paper UI" comentada anteriormente. Simplemente hay que acceder a través de un navegador desde cualquier equipo conectado a la misma red que el sistema y en el menú de la izquierda elegir la opción de "*Add-ons*" y navegar entre las distintas categorías o utilizando el buscador hasta dar con la funcionalidad que buscamos.

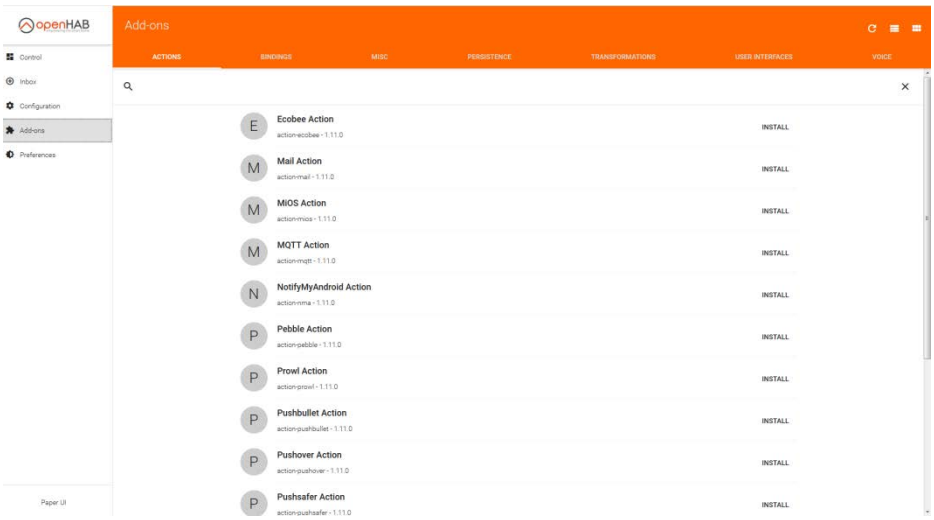

Figura 11. Interfaz "Paper UI" desde donde se pueden añadir módulos al sistema openHAB.

Opcionalmente también se puede descargar el archivo "*.jar*" del módulo a añadir y depositarlo en la carpeta correspondiente del sistema, de modo que en el siguiente reinicio se habilitará también para su utilización. Este método es útil para instalar paquetes en desarrollo (versiones beta) que todavía no han sido aprobados oficialmente, o versiones anteriores a las más actuales (si existe un problema de compatibilidad que antes no se daba, por ejemplo), que no aparecen en los listados del "Paper UI".

4.3. MANTENIMIENTO Y SEGURIDAD

El mantenimiento del sistema resulta poco exigente, dado que una vez está funcionando correctamente resulta recomendable aplicar la máxima de "no arreglar lo que no está roto". Podría darse el caso de que al actualizar los paquetes instalados se introduzcan cambios de configuración, nuevas definiciones de los objetos, modificaciones en la sintaxis, etc. que pueden acarrear que algún elemento de nuestra instalación deje de funcionar, o incluso en el peor de los casos, que todo el sistema se vuelva inusable.

La recomendación es limitar estas actualizaciones a un par de veces al año, y siempre realizando el proceso habitual de copia de seguridad antes: primero apagar el sistema, extraer la tarjeta SD y guardar una imagen en una ubicación segura. Después se puede volver a encender, actualizar todo lo que sea necesario y verificar que todo funcione correctamente. Incluso si todo estuviera bien es una buena práctica guardar esa imagen de la tarjeta con un sistema funcionando bien (aunque esté desactualizado), por si en el futuro se presentara algún problema y fuera necesario volver a ese momento anterior.

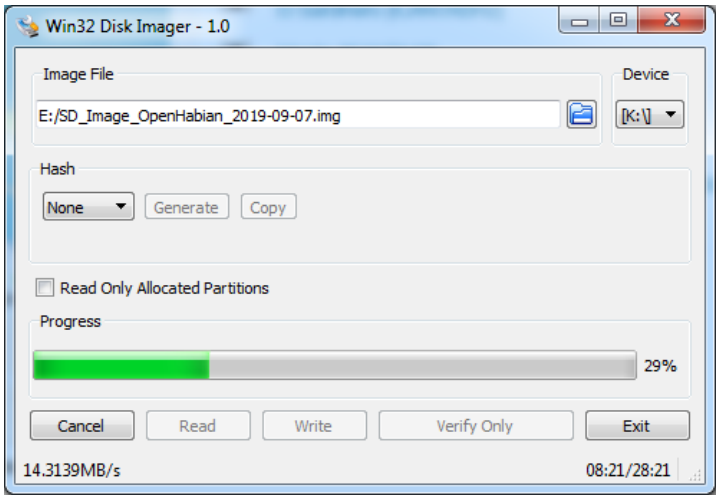

Figura 12. Proceso de copia de seguridad: copiado íntegro de la tarjeta SD a un archivo de imagen ".img*"*

En cuanto a la seguridad, se podría argumentar que la práctica de no aplicar los parches de este tipo en el sistema operativo durante mucho tiempo puede suponer una potencial brecha en nuestro equipo. En general sería así, pero hay que tener en cuenta que lo más habitual es que la máquina no pueda ser accesible de ninguna manera desde Internet, dado que por un lado está ubicada detrás del *router* local y su correspondiente cortafuegos, y por otro no tiene expuesto ningún puerto ni servicio que pueda ser atacado desde el exterior. Esto es matizable si se instalan servicios adicionales en la misma máquina, y por tanto la opción de mantener una Raspberry Pi dedicada exclusivamente a operar como controlador central del sistema domótico es recomendable.

4.3.1. ADMINISTRACIÓN AVANZADA: LA CONSOLA

Si bien escapa del ámbito del presente proyecto por tratarse de funcionalidad compleja y que requeriría bastante extensión para ser explicada, se quiere comentar la existencia de la consola *Karaf* de Apache[25]. *Karaf* es un moderno entorno de desarrollo Java para aplicaciones

[25] http://karaf.apache.org/

empresariales desplegables en la nube o en máquinas propias. La consola, integrada con la distribución de openHAB, es una potente herramienta que permite interactuar con el núcleo de la instalación del sistema para realizar tareas de otro modo inaccesibles.

Se accede a la consola desde el sistema openHABian, tras conectarse por SSH como se ha visto ya anteriormente, mediante el comando "*openhab-cli console*". Para la cuenta "openhab" la contraseña por defecto es "habopen". Se abandona la consola ejecutando la orden "*logout*".

Algunas de las funcionalidades avanzadas son:

- Lanzamiento de **comandos en tiempo de ejecución**, de forma que se puede consultar o incluso variar el estado de entidades como los ítems o cosas. Algunas de estas órdenes se ejecutan directamente sobre la base de datos interna del sistema y pueden llegar a desestabilizarlo o incluso hacer que deje de funcionar, por lo que deben utilizarse con cautela.
- **Gestión de paquetes**, siendo posible ver qué está instalado en el sistema, parar y arrancar módulos, lo que puede resultar útil a la hora de detectar y corregir fallos desde un acceso de bajo nivel.
- **Acceso a los registros** ("*logs*") del sistema en tiempo real, con la posibilidad de filtrar lo que se muestra en la consola, borrar registros, o cambiar el nivel de log para que se guarden eventos de más o menos importancia (error, aviso, información, depuración, traza, todos o apagado).

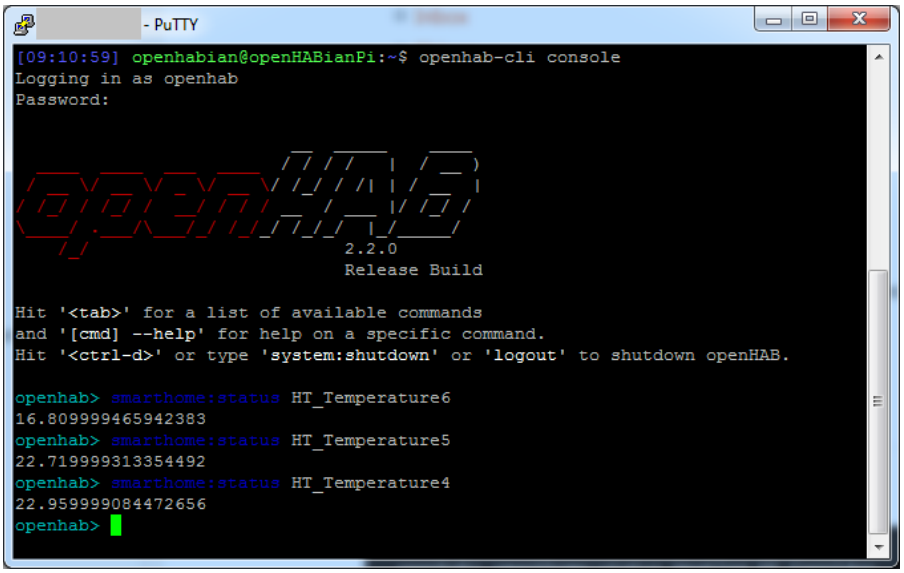

Figura 13. Acceso a la consola Karaf y ejemplos de comandos para consultar el estado de tres sensores de temperatura de la instalación.

El acceso al Log del sistema también puede realizarse desde otros lugares, como por ejemplo a través del "logtail" disponible desde un navegador en la dirección comentada ya en páginas anteriores. La configuración de los mismos también se puede modificar, además de desde la consola, editando el archivo que recoge los parámetros que gobiernan qué se registra y cuándo[26]. Este archivo se encuentra en la ruta "/var/lib/openhab2/etc", y tiene el nombre por defecto "*org.ops4j.pax.logging.cfg*".

[26] https://www.openhab.org/docs/administration/logging.html

También resulta interesante la opción de publicar mensajes “bajo demanda” en los “logs”, de modo que se puede trazar el funcionamiento (correcto o incorrecto) del sistema verificando que las situaciones, valores, etc. se correspondan con los esperados, o que los eventos se produzcan y/o desencadenen como está previsto. En los anexos se puede ver cómo se integran este tipo de mensajes en las definiciones de las reglas de automatización.

Por último, en este apartado de administración, cabe destacar que en algunas ocasiones es necesario reiniciar la plataforma openHAB para que se apliquen cambios en la configuración, lo cual se puede hacer simplemente utilizando el siguiente comando desde una consola del sistema operativo:

```
sudo systemctl restart openhab2.service
```

5. DESCRIPCIÓN DE LOS MÓDULOS Y ELEMENTOS DE LA INSTALACIÓN

Este capítulo pretende realizar una descripción general de los distintos módulos y elementos incorporados en el despliegue domótico realizado, tanto a nivel software como hardware, explicando brevemente su finalidad, configuración y comportamiento.

Se ha realizado una categorización basada en la funcionalidad desempeñada por cada subsistema en el conjunto dado que puede dar una visión de la instalación más accesible para cualquier lector. Pero también sería posible una organización basada en los distintos tipos de dispositivos (sensores, actuadores, concentradores), en la tecnología de interconexión que utilizan (Bluetooth, Wi-Fi, ZigBee), su marca comercial (Itead, Philips, Xiaomi), etc.

5.1. MÓDULOS GENERALES MULTIPROPÓSITO

En esta categoría se engloban las funcionalidades que tienen un propósito general, habitualmente para dotar de mayor potencia otras áreas de la instalación o realizar tareas transversales a cualquier área funcional como la comunicación, la verificación de conectividad o la sincronización horaria.

El orden de presentación no es indicativo de mayor o menor relevancia, dado que todos son importantes, y obedece a un criterio estrictamente alfabético.

5.1.1. MAIL ACTIONS

El módulo de correo electrónico permite al sistema el envío de mensajes a través de este medio a cualquier cuenta.

Es recomendable crear una cuenta de correo específica para utilizar de manera exclusiva con este servicio, definiendo además credenciales de acceso distintas de las que se usen para otros servicios de Internet. De este modo se evita el riesgo de que pueda llegar a interferir con una cuenta destinada a un uso más tradicional y habitual, aunque sería perfectamente viable utilizar una ya existente de este tipo.

La instalación se realiza desde el “Paper UI”. Una vez instalado se puede configurar editando el archivo de configuración llamado “*mail.cfg*” ubicado en la carpeta “*services*”. Los parámetros necesarios serán:

- **Hostname**: el nombre del servicio de correo que se va a utilizar. Por ejemplo, para Gmail es “smtp.gmail.com”.
- **Port**: el puerto de comunicación a utilizar para conectar con el servicio. Por defecto es el 25 (SMTP[27] estándar), pero es recomendable e incluso necesario según el servicio elegido que se utilice un puerto seguro como el 587 (TLS/SSL[28]).
- Las **credenciales** de acceso a la cuenta: nombre de usuario y contraseña.
- La **dirección de correo** desde la que se mandarán los mensajes (que corresponderá, lógicamente, a las credenciales introducidas anteriormente).

[27] Simple Mail Transfer Protocol.

[28] Transfer Layer Security/Secure Sockets Layer.

El resto de configuraciones, que están disponibles en la documentación oficial del módulo[29], pueden ser necesarias o no en función del proveedor de correo electrónico elegido.

Una vez configurado es muy sencillo utilizar la funcionalidad disponible, ya sea a través de "Reglas" o de secuencias de comandos ("*scripts*"). Basta con definir la siguiente acción a realizar, donde los parámetros resultan auto explicativos:

```
sendMail ("Dirección_destino", "Asunto", "Cuerpo_mensaje")
```

Las posibles utilidades son numerosas, pero en la instalación objeto de este estudio se emplea para enviar una notificación a una hora predeterminada cuando algún elemento de la instalación no ha cambiado su estado en más de 24 horas, lo que podría indicar que está desconectado de la red, se ha quedado sin batería o presenta una avería.

Se ha considerado realizar un informe semanal o mensual sobre el estado de las baterías de los distintos dispositivos que dan información al respecto, de modo que se pudiera planificar un reemplazo preventivo, pero se ha constatado que la fiabilidad de esa información no siempre es elevada, por lo que se ha desechado la idea.

También podría utilizarse como sistema de notificaciones para el módulo de seguridad, pero en este caso sería más como secundario de respaldo al principal que se comentará un poco más adelante, dado que un correo no siempre es algo de lo que el usuario pueda enterarse de forma inmediata.

5.1.2. MQTT BINDING

"MQTT significa MQ Telemetry Transport. Se trata de un protocolo de mensajería de publicación/suscripción extremadamente simple y ligero, diseñado para dispositivos limitados y redes con bajo ancho de banda, elevada latencia o poco fiables. Los principios de diseño son minimizar el ancho de banda y los requisitos del dispositivo mientras se intenta también garantizar la confiabilidad y cierto grado de seguridad de entrega. Estos principios resulta que convierten el protocolo en ideal para el emergente mundo de dispositivos conectados M2M[30]/IoT, y para aplicaciones móviles donde el ancho de banda y el consume de batería son demandados." [3]

En definitiva, una manera sencilla y eficiente de intercambiar mensajes (comandos) entre el controlador central (openHAB) hacia/desde los dispositivos conectados, siendo todos ellos clientes del sistema MQTT, incluso el propio sistema openHAB.

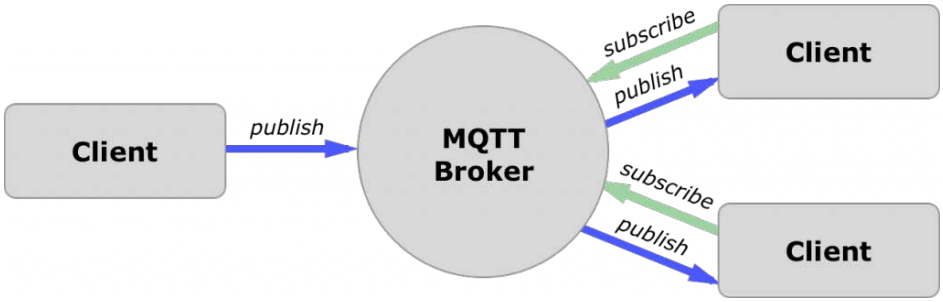

Figura 14. Esquema de la comunicación en una red MQTT (fuente).

[29] https://www.openhab.org/addons/actions/mail/

[30] Machine to (two) Machine, de máquina a máquina.

Esta unión o "*binding*" no incluye un servidor o gestor de mensajes (bróker), sólo los canales para establecer las comunicaciones entre los distintos clientes. Para la instalación bajo estudio se ha usado uno de código abierto de los más extendidos: *Mosquitto*[31]. Se ha instalado en la misma máquina dado que no supone apenas carga para el hardware, y la configuración se hace a través del un archivo de texto con nombre "*mqtt.cfg*" que está ubicado en la carpeta de configuración "*services*".

El único parámetro a configurar obligatoriamente es "*broker.url*", la dirección donde está instalado *Mosquitto*, indicando también el puerto en el que está escuchando – lo que a su vez definirá si la conexión es segura (si se fija el puerto SSL) o no. Opcionalmente se puede fijar la obligatoriedad de que los mensajes incorporen credenciales de nombre de usuario y contraseña, que no tiene razón de ser en este entorno, así como otras configuraciones relativas a la calidad del servicio (QoS), la sincronía de los mensajes o el comportamiento que debe presentar al perder la conexión, entre otras[32].

Una vez configurado el bróker, con el MQTT "*binding*" ya instalado, es posible enviar (publicación) o recibir (suscripción) información entre el bróker y cualquier dispositivo que esté conectado a la red y sea compatible con éste protocolo. La información está organizada en temas ("*topics*"), de modo que se puede seleccionar qué mensajes son relevantes para cada dispositivo. Es necesario definir en cada dispositivo su nombre (incluyendo si pertenece a una categoría), generalmente en el apartado de configuración MQTT. Los temas pueden ser de dos tipos:

- **Comandos**, en los que el dispositivo puede recibir mensajes que cambian su estado. Por ejemplo, un relé llamado "TH1601" que esté dentro de la categoría "sonoff" y que presenta un elemento controlable llamado "relay/0" se puede encender si recibe un comando ":state:ON:1", o apagar si recibe ":state:OFF:0".
- **Estados**, donde se puede consultar información sobre la situación actual del dispositivo. Por ejemplo, el estado del mismo relé comentado antes se podrá consultar en "/sonoff/TH1601/relay/0:state".

La estructura de los mensajes entrantes y salientes puede resultar confusa y engorrosa al principio, pero tras un par de intentos la filosofía de comunicación se hace evidente, siendo sencillo extender la funcionalidad a muchas de las áreas de automatización.

5.1.3. NETWORK BINDING

Permite comprobar si un dispositivo está disponible en la red mediante dos métodos: haciéndole "*ping*"[33] o estableciendo con éxito una conexión TCP en un puerto especificado. De este modo se pueden establecer rutinas que avisen al administrador/usuario de la falta de conectividad de alguno de los elementos de la instalación a través de alguno de los módulos de comunicación, como por ejemplo el correo electrónico que se ha visto con anterioridad.

[31] https://mosquitto.org/

[32] https://www.openhab.org/addons/bindings/mqtt/

[33] Utilidad de administración de redes que permite verificar la disponibilidad de un equipo en una red IP.

La configuración del módulo resulta bastante sencilla: basta con crear un archivo de texto en la carpeta de servicios, al igual que en los anteriores, con el nombre "*network.cfg*". Las opciones que pueden fijarse son:

- **allowSystemPings**: permite utilizar la herramienta de "ping" del sistema, en lugar de usar la interna del servidor Java, que puede no ser compatible con todos los dispositivos. El valor por defecto es "*true*", activada.
- **allowDHCPlisten**: para localizar las entradas y salidas de la red de equipos que obtienen su dirección IP mediante este mecanismo DHCP[34], de modo que la detección de conexiones y desconexiones de los mismos es más rápida. Útil por ejemplo para teléfonos inteligentes y tabletas. El valor por defecto es "*true*".
- **arpPingToolPath**: ubicación de la herramienta "*arping*", similar en funcionalidad al "ping" pero operando sobre protocolos de red diferentes. El valor por defecto es "*arping*".
- **cacheDeviceStateTimeInMS**: el resultado de la detección de disponibilidad de red almacena temporalmente durante un breve periodo de tiempo fijado por esta configuración, en milisegundos. En ese intervalo no se generan nuevas peticiones de "ping". El valor por defecto es 2000 (2 segundos).

Para poner en uso la funcionalidad instalada es preciso primero generar un canal a través de una definición de una "Cosa" en el archivo "*network.things*", ubicado en la carpeta de configuración correspondiente y que permitirá agrupar todos los elementos de este tipo. En esa definición se dará un nombre al dispositivo a verificar y se especificará su dirección de red (parámetro "*hostname*"), normalmente a través de su IP. Opcionalmente, se podrá establecer el número de reintentos ("*retry*", 1 por defecto) de "ping" que se efectuarán, el tiempo de espera máximo en milisegundos ("*timeout*", 5000 por defecto) y el intervalo de refresco ("*refreshInterval*", 60000 por defecto) también en milisegundos, que marcará la frecuencia con la que se consultará el estado del dispositivo. Un ejemplo de definición de un canal usado en el sistema es:

```
network:pingdevice:RaspiServiping [ hostname="X.X.X.X", retry=1, timeout=5000, refreshInterval=60000 ]
```

Con el canal ya definido, el siguiente paso es configurar un "Ítem" asociado al mismo, que será el que varíe su estado en función de la información recogida en el canal, y que permite asociar lógica de automatización. Una posibilidad es definirlo como un interruptor, "*Switch*", que cambiará su valor a encendido/ON o apagado/OFF en función de si el dispositivo asociado está disponible en la red o no. La definición, ubicada en un archivo de ítems específico para elementos de red de nombre "*network.items*", es:

```
Switch RaspiServiping { channel="network:pingdevice:RaspiServiping:online" }
```

Y el estado del dispositivo en la red será el que se almacene en "*RaspiServiping.state*": ON si responde al ping enviado, OFF si no lo hace tras el tiempo de espera y reintentos configurados.

En el caso práctico de este trabajo la información de los diversos dispositivos que permiten consultar su estado de red se notifica una vez al día a través de un mensaje de correo electrónico, a una hora determinada. Si algún dispositivo no está conectado se pueden tomar

[34] *Dynamic Host Configuration Protocol*: protocolo de red en el que un servidor asigna direcciones IP dinámicamente a los dispositivos que soliciten conectarse a la misma.

medidas correctivas. También se ha probado a mandar los avisos a medida que se detectan las desconexiones, pero dado que ninguno de los elementos a supervisar es especialmente crítico, y que cuando pierden la conexión (algo poco habitual) no suelen recuperarla por sí mismos y hay que reiniciarlos, se ha preferido mantener la opción de suministrar un único informe diario. Si se tratara de elementos críticos se podría incluso cambiar el método de notificación para que se realizara a través de la mensajería instantánea, opción que se trata más adelante en este documento.

5.1.4. NTP BINDING

Este modulo proporciona una funcionalidad sencilla pero potente: sincronizar la hora del sistema con la de algún servidor NTP[35], el protocolo de Internet para sincronizar los relojes de distintos dispositivos a través de la red. Es una manera sencilla de disponer de la fecha y hora actuales en la zona horaria correspondiente de cara a realizar transformaciones, operaciones o registros de eventos.

Tras instalar el "*binding*", que no requiere ninguna configuración, basta con añadir una "Cosa" para disponer de la información que proporciona en el canal "*dateTime*". Por ejemplo, en el entorno descrito la definición en el archivo "*ntp.things*" es:

```
ntp:ntp:demo  [ hostname="hora.rediris.es", refreshInterval=60, refreshNtp=30 ]
```

Donde "*hostname*" es el servidor NTP al que se desee conectar, que en este caso forma parte de RedIRIS[36]; "*refreshInterval*" es el tiempo entre consultas al servidor, en segundos; y "*refreshNtp*" el número de actualizaciones publicadas en el canal entre consultas al servidor NTP. A partir de aquí se puede definir un ítem de tipo "*DateTime*" (fecha y hora) que almacene la información obtenida del servidor a través del canal establecido. A modo de ejemplo, en el archivo "*ntp.items*":

```
DateTime Date "Date [%1$tA, %1$td.%1$tm.%1$tY %1$tH:%1$tM]" { channel="ntp:ntp:demo:dateTime" }
```

Esta expresión incluye una transformación de los datos[37], expresada entre los corchetes [ ], de modo que el formato almacenado en "*Date*" se pueda percibir claramente como una fecha: "*DíaSemana DíaMesAño Hora:Minutos*".

5.1.5. OPENHAB CLOUD CONNECTOR

El conector con la nube de openHAB permite acceder a la instalación local del entorno de control domótico desde una ubicación remota a través de una nube, habitualmente la alojada por la openHAB Foundation: **myopenHAB.org**[38]. De este modo se pueden controlar los dispositivos como si se estuviera conectado a la red doméstica, lo cual no es posible de otro modo.

Sin embargo esta opción presenta una potencial brecha en la seguridad y confidencialidad de la instalación, dado que al depender de un servicio externo no resulta posible garantizar quién

---

[35] Network Time Protocol

[36] https://www.rediris.es/rediris/

[37] https://www.openhab.org/docs/configuration/transformations.html

[38] http://www.myopenhab.org/

tiene acceso al mismo (por ejemplo, administradores de la nube), no somos los únicos poseedores de las credenciales (la nube puede sufrir un ataque, y resultar filtrados), y se cede un registro de la actividad de los usuarios y los dispositivos, una información que sin duda es valiosa y permite mantener el servicio como "gratuito".

Por estos motivos, para la solución adaptada en la vivienda no es una funcionalidad que esté instalada. En su lugar se ha construido un método alternativo de acceso desde el exterior a los servicios alojados en la red local que será tratado en el siguiente capítulo de este trabajo.

5.1.6. TELEGRAM ACTIONS

El último módulo de propósito general añadido al sistema es otro método de comunicación con los usuarios y/o administradores de la plataforma, basado en este caso en el servicio de mensajería ***Telegram***[39]. Es una alternativa al archiconocido *WhatsApp*, presentando un funcionamiento muy similar pero con algunas ventajas en el aspecto de la seguridad y las posibilidades de integración con otros entornos.

La instalación del módulo en el entorno de openHAB es sencilla a través de la "Paper UI", pero su configuración y puesta en marcha es algo más compleja que en los ejemplos anteriores porque involucra operaciones tanto en el entorno local como en los servidores de Telegram, por lo que es necesario tener previamente una cuenta activa en este servicio de mensajería.

En la documentación oficial del módulo[40] se pueden encontrar los prerrequisitos necesarios para poder conectar openHAB con la API de Telegram, y a modo de resumen son:

- Crear un nuevo "*bot*"[41], abriendo una nueva conversación con el "*BotFather*" de Telegram, y obtener el "*Token*" (un conjunto de caracteres) de autenticación rellenando la información que solicita el comando "*/newbot*".
- Abrir un nuevo chat con el bot creado en el apartado anterior, y mandarle un mensaje. Es necesario que el chat tenga al menos un mensaje para poder completar el siguiente paso.
- Obtener el "*chatId*", el número de identificación del chat creado anteriormente. En un navegador web hay que abrir la dirección:

  https://api.telegram.org/bot<token>/getUpdates

  Donde <*token*> es el valor obtenido anteriormente. En el JSON resultante hay que localizar el valor de "*id*", que es el "*chatId*" buscado. Será un valor numérico precedido de un guión.

Con esta información ya es posible configurar el módulo en openHAB editando el archivo "*services/telegram.cfg*" con los siguientes parámetros requeridos:

- **bots**: un listado con el nombre (*botName*) de los robots configurados para trabajar con la plataforma, separados por una coma.
- **botName.chatId**: el identificador del chat para el bot dado.
- **botName.token**: el "*token*" de autenticación para el "*bot*" en cuestión.

---

[39] https://telegram.org/

[40] https://www.openhab.org/addons/actions/telegram/

[41] Aféresis de robot, es un programa que realiza tareas de manera autónoma, pudiendo interactuar con usuarios y sistemas.

A partir de este punto ya es posible definir acciones en las reglas del sistema, de modo que cuando se produzca algún evento relevante (apertura de un sensor de puertas o ventanas, activación de un sensor de presencia, un determinado valor en un sensor de temperatura, la pérdida de conectividad de un elemento de la instalación, etc.) se notifique a los usuarios o al administrador. También es posible adjuntar una imagen al mensaje enviado.

En la instalación objeto de este estudio se utiliza principalmente para las notificaciones del módulo de seguridad. Aunque más adelante se estudiará este módulo con más detenimiento, a continuación se muestra un ejemplo de una regla completa basada en un sensor de contacto magnético para una puerta. El texto detrás de los caracteres "//" es un comentario en el código.

```
rule "Alarma puerta principal" //Nombre de la regla
when //Cuando el estado del sensor cambia a "abierto"
        Item WindowSwitch1_Status changed to OPEN
then //Entonces, si la alarma está activada, enviar el mensaje de Telegram
        if(AlarmOnOff.state == ON){
        sendTelegram("OHR8bot", "Puerta del GARAJE abierta")
        }
end
```

En las siguientes imágenes se puede ver la ubicación del sensor en la puerta principal, de la marca Xiaomi (Figura 15), y una captura de pantalla del chat de notificaciones en Telegram para el sistema de alarma (Figura 16).

Figura 15. Sensor magnético de contacto ubicado en la puerta principal de la vivienda.

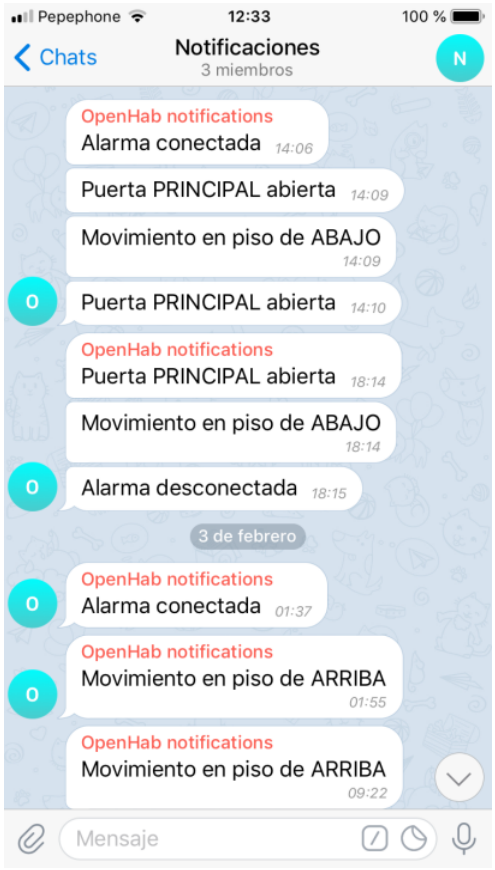

Figura 16. Captura de pantalla del chat de notificaciones en la aplicación Telegram, en un iPhone.

5.2. MÓDULOS DE PERSISTENCIA DE ESTADO

Los componentes estudiados en este subapartado se encargan de gestionar el almacenamiento de la información para, principalmente, garantizar la persistencia de estado de los dispositivos de la instalación.

La persistencia consiste en almacenar en una base de datos en qué estado está cada dispositivo. Se puede realizar siguiendo diversas estrategias: periódicamente (cada determinados horas, minutos o segundos), siempre que haya un cambio de estado, en función del resultado de un script, etc. Y el almacenamiento puede ser en la base de datos interna de openHAB, que no posee persistencia (solo almacena el estado actual), o en una base de datos externa que, a su vez, puede estar alojada en la misma máquina que ejecuta el sistema de automatización o en otra ubicación.

En la instalación estudiada se utilizan dos bases de datos externas para las estrategias de persistencia: una instalada en la misma Raspberry que el openHABian, el motor RRD4J; y otra ubicada en otra máquina conectada a la misma red (ambas Raspberry están ubicadas juntas en la misma caja empotrada), el motor InfluxDB. Las dos podrían correr sin problemas en la misma máquina, pero dado que la funcionalidad de InfluxDB es avanzada, como se verá a continuación, se eligió instalarla en otro equipo para poder trastear con ella sin miedo a impactar negativamente al funcionamiento del núcleo del sistema.

5.2.1. INFLUXDB PERSISTENCE

"InfluxDB es una base de datos de series de tiempo construida desde cero para manejar altas cargas de escritura y consultas […]. InfluxDB está destinado a ser utilizado como una almacenamiento de reserva para cualquier tipo de uso que involucre grandes cantidades de datos con marcas de tiempo, incluyendo la supervisión DevOps, métricas de aplicaciones, datos de sensores IoT y análisis en tiempo real." [4]

El uso en la gestión de los datos de un entorno domótico es, por tanto, una aplicación perfecta para este motor de base de datos, dado que todos los eventos se registran con su "*timestamp*", un número de secuencia (identificador único del evento) y el valor registrado. De este modo toda la actividad que se desee se guarda de manera eficiente y sin compresión ni reducción ("*trimming*") para su posterior consulta, análisis o representación gráfica.

El prerrequisito para utilizarla en conjunción con openHAB es tener un servidor de InfluxDB instalado, lo cual es bastante sencillo siguiendo los tutoriales de instalación y configuración disponibles en la documentación oficial[42]. Después basta con crear una base de datos (llamada "openhab", por defecto) y un usuario con permiso de escritura (también "openhab", por defecto), al que se le asigna una contraseña que usaremos más adelante.

A continuación se puede configurar ya el servicio, editando el archivo "*services/influxdb.cfg*" con las siguientes propiedades:

- **url**: ubicación del servidor InfluxDB. Si no se especifica ninguno, el valor por defecto es la dirección de "*localhost*"[43]: *http://127.0.0.1:8086*.

[42] https://docs.influxdata.com/influxdb/v1.7/introduction/installation/

[43] El ordenador o dispositivo local que se está usando, que siempre tiene asignada la dirección 127.0.0.1.

- **user**: el nombre de usuario con los permisos necesarios, tal y como se ha descrito anteriormente. El valor por defecto es "openhab".
- **password**: contraseña para el usuario.
- **db**: nombre de la base de datos creada en el servidor. Por defecto es "openhab".

Finalmente, con todos los prerrequisitos y configuraciones ya listas, se puede empezar a guardar información en la base de datos definiendo la estrategia o estrategias a seguir y sobre qué elementos.

En la instalación de este trabajo se ha elegido guardar en esta base de datos todos los cambios de estado de todos los dispositivos, y también su estado una vez al día (en el supuesto de que no cambien más veces). Además, se ha configurado para que en un reinicio del sistema el estado de cada uno de los dispositivos se restaure a su último valor conocido, para que un corte en el servicio, independientemente de que sea programado o imprevisto, no altere el funcionamiento del mismo. El código para esto es:

```
* : strategy = everyChange, everyDay, restoreOnStartup
```

Donde el "*" indica que la estrategia debe aplicarse sobre todos los ítems.

Asimismo, es posible establecer estrategias adicionales para tener registros más completos, como por ejemplo para valores que vayan a ser representados de forma gráfica y que no cambien su estado con la frecuencia suficiente para que esas representaciones resulten vistosas. A modo de ejemplo, la siguiente estrategia define que el valor de una serie de ítems relacionados con la temperatura y el control de la calefacción se guarde cada minuto:

```
TH1601_temp, GF_Heating_TargetTemp, HT_Temperature6, ProxySonOff_esp01: strategy = everyMinute
```

El primero es un sensor de temperatura (de Itead), ubicado en el salón; el segundo, el valor objetivo de la temperatura en la planta baja; el tercero es otro sensor de temperatura (de Xiaomi), ubicado en el exterior de la vivienda; y el cuarto es el interruptor que controla un relé Wi-Fi (también de Itead) encargado de encender o apagar el circuito de calefacción en la planta baja.

5.2.2. RRD4J PERSISTENCE

Este módulo de persistencia, que está instalado por defecto en la distribución estándar de openHAB, funciona sobre una base de datos "*Round-Robin*"[44]. RRD4J básicamente significa "*Round Robin Database for Java*", base de datos RR para Java.

A diferencia de otras bases de datos típicas, una base de datos RR no aumenta su tamaño con el tiempo, sino que se ciñe al que se le marca en la configuración inicial mediante compresión de datos y reciclaje del espacio guardando solo ciertos conjuntos de los mismos. Los datos más nuevos poseen una granularidad mayor, mientras que los antiguos tendrán muestras más distanciadas: es posible que para el valor de un sensor se guarden datos a cada minuto de las últimas 24 horas, pero que solo existan registros diarios para el último año.

La información almacenada no se puede consultar directamente, aunque sí puede usarse para restauraciones de estado tras un reinicio o para representaciones gráficas.

[44] https://aggregate.tibbo.com/es/technology/storage/round-robin-databases.html

Pero la compresión de los datos le resta bastante utilidad en un sistema donde las series cronológicas de información completas pueden resultar muy relevantes para la toma de decisiones: comparación de temperaturas entre distintos meses a lo largo de varios años, identificación de patrones de consumo repetitivos, elección de las estrategias óptimas de climatización, etc.

Por este motivo, pese a que es un módulo muy bien documentado y ampliamente usado, en el caso del sistema bajo estudio se ha decidido no usarlo para ninguna aplicación en concreto, aunque no se ha deshabilitado para tener un respaldo al menos de los datos más recientes.

5.3. MÓDULO DE SEGURIDAD

Una de las aplicaciones más interesantes del sistema estudiado es la construcción de un sistema de seguridad doméstica a medida de la vivienda, sin apenas mantenimiento y sin costes recurrentes. Es cierto que no se dispone del respaldo de una empresa de seguridad conectada al mismo, pero para verificaciones rutinarias, evitar olvidos y agregar algo de tranquilidad resulta suficiente. Además se garantiza la privacidad, dado que no hay posibilidad de que alguien esté accediendo a los datos del sistema por estar este controlado únicamente por los habitantes de la casa.

Este subapartado se centra en la descripción de los elementos que componen el módulo de seguridad que está basado en componentes de la marca china Xiaomi[45], tanto a nivel hardware como su configuración y funcionamiento en openHAB.

(sensores en puertas y ventanas; sensores de presencia; configuración de alarma basada en Telegram; sistema Xiaomi aislado, introducción a cómo modificarlo).

5.3.1. XIAOMI MI SMART HOME BINDING

El kit de domótica lanzado por la marca china hace un par de años supuso una auténtica revolución por su precio contenido, amplias funcionalidades y consumo reducido de sus sensores que les dota de largas autonomías (más de un año sin necesidad de cambiar las pilas). Además se trata de un conjunto escalable al que se le pueden añadir más elementos, siendo por tanto personalizable para las necesidades de cualquier vivienda.

El conjunto original está compuesto por:

- **Puerta de enlace** o "**Gateway**": la pieza central del subsistema. Se conecta a la red WiFi doméstica, y hace de enlace con el resto de sensores y actuadores del kit que se conectan a él por una versión particular del protocolo ZigBee[46] [5]. Además incorpora un altavoz para reproducir sonidos, y una lámpara LED multicolor.
- **Enchufe inteligente**: sin aplicación directa para el subsistema de seguridad (ver siguiente apartado de iluminación).
- **Interruptor sin cables**: podría emplearse para activar o desactivar el sistema de alarma. Actualmente está programado para encender y apagar un enchufe inteligente (ver siguiente apartado).
- **Sensor de temperatura y humedad**.
- **Sensor de puerta** o **ventana**: un sensor magnético, compuesto por dos piezas, que detecta cuándo una se separa de la otra.

[45] https://www.mi.com/global/about

[46] Protocolo de comunicación inalámbrico (EEE 802.15.4) de bajo consumo que opera en la banda de 858Mhz. (en Europa), pensado para aplicaciones en el hogar como la seguridad y la automatización.

Figura 17. A la izquierda, componentes del kit de domótica de Xiaomi, con el Gateway en el centro. A la derecha, un sensor de temperatura, humedad y presión atmosférica, y un sensor de movimiento.

A estos elementos se han ido añadiendo paulatinamente otros, hasta completar el ecosistema actual compuesto, además del Gateway, por:

- 2 **sensores de movimiento** por infrarrojo (IR), uno de ellos con detección de luminosidad. Ubicados estratégicamente en cada una de las plantas de la vivienda.
- 3 sensores de puertas y ventanas adicionales, siendo un total de 4 para controlar el estado todas las puertas accesibles de la residencia.

La integración del kit Xiaomi es una de las más laboriosas a realizar, aunque no puede llegar a considerarse complicada. Tras instalar el "*binding*" a través del "Paper UI", como en el resto de extensiones anteriores, es necesario conectar el Gateway a la red WiFi local a través de la aplicación "*MiHome*", siguiendo el proceso descrito por el fabricante y seleccionando como ubicación la región "Mainland (China)", dado que de lo contrario no estarán disponibles algunas de las opciones necesarias. Por este mismo motivo, también es importante no actualizar el firmware a su última versión aunque la aplicación de Xiaomi lo solicite.

A continuación hay que poner el Gateway en modo de desarrollador ("*developer mode*"), que es el paso que puede resultar más complejo. Hay que ir al apartado "About" de la aplicación "*MiHome*" y tocar repetidamente en el número de versión que aparece en la parte inferior hasta que se habilite el modo de desarrollador. Entonces deberían aparecer dos opciones adicionales: la de "*wireless communication protocol*" que activaremos para habilitar las funcionalidades WiFi del dispositivo; y la de "*hub info*", donde estará la clave de desarrollador única del dispositivo. Esta clave o "*key*" es la que debe suministrarse en la configuración de la "*Thing*" correspondiente al Gateway, y que dará acceso no solo a la funcionalidad del mismo sino a todos los dispositivos que dependan de él y sean compatibles con el "*Binding*" de openHAB.

Con la funcionalidad WiFi activada el dispositivo ya puede ser conectado a openHAB, siendo lo más cómodo "descubrirlo" a través de la funcionalidad de la "Paper UI". A partir de este punto el Gateway ya no necesita conexión a Internet salvo que queramos acceder a la información que gestiona desde la propia aplicación. Por eso se puede bloquear todo el tráfico saliente el "*router*" que gestiona las conexiones en la vivienda para la IP asignada al Gateway, de modo que se garantiza la privacidad de todos los datos que fluyen a través del mismo.

Los distintos dispositivos periféricos se pueden incorporar al sistema a través de la aplicación (salvo si el tráfico está bloqueado, aunque podría habilitarse para conectarlo y volver a cortarlo

después), de manera manual utilizando la funcionalidad del propio Gateway (pulsando su botón 3 veces seguidas) o directamente desde el "*Binding*" de openHAB (forzando una secuencia de descubrimiento). Una vez que el dispositivo está conectado al Gateway aparecerá en el apartado "*Inbox*" de la "Paper UI", y puede configurarse desde ese asistente o anotar su identificador de canal para hacerlo a través de archivos de configuración manuales (que es la opción empleada para la instalación bajo estudio).

A partir de este punto la configuración del sistema resulta bastante sencilla. Los detectores de movimiento cambian su estado a "ON" cuando se activan, y pasan a "OFF" un minuto después; este comportamiento no puede personalizarse, lo cual es un punto débil porque no se pueden tener muestreos en intervalos menores a un minuto, pero es una elección de diseño del fabricante para mantener un consumo bajo de batería (se garantiza que solo se reporta un movimiento por minuto). Los sensores de las puertas cambian su estado a "OPEN" o "CLOSED" en función de si las dos partes están enfrentadas y próximas o no. Los dos tipos de dispositivos proporcionan también, además del nivel de batería (que no resulta demasiado fiable, por cierto), el último momento en que se han activado mediante los valores de "*LastMotion*" y "*LastOpen*", respectivamente. A continuación se puede ver ejemplos del código de definición de un elemento de cada tipo:

```
//Sensor de movimiento del hall
Switch MotionSensorHall_MotionStatus <motion> {
channel="mihome:sensor_motion:id:motion" }

DateTime MotionSensorHall_LastMotion "[%1$tY-%1$tm-%1$td %1$tH:%1$tM:%1$tS]" <clock-
on> { channel="mihome:sensor_motion:id:lastMotion" }

//Sensor de la puerta principal
Contact WindowSwitch1_Status "Entrada [%s]" <frontdoor>  (Contacts) {
channel="mihome:sensor_magnet:id:isOpen" }

DateTime WindowSwitch1_LastOpened "[%1$tY-%1$tm-%1$td  %1$tH:%1$tM:%1$tS]" <clock-
on> { channel="mihome:sensor_magnet:id:lastOpened" }
```

La rutina de control de la alarma, vinculada al programa de mensajería Telegram como se ha visto anteriormente, está desarrollada a través de reglas de automatización en el archivo "*telegram.rules*". Básicamente lo que se hace es responder a cada cambio en cualquiera de los sensores de presencia o de las puertas, y avisar mediante un mensaje en el chat de notificaciones si la alarma está conectada. Por ejemplo, para el sensor definido anteriormente la regla asociada es:

```
rule "Alarma movimiento en el hall"
when
        Item MotionSensorHall_MotionStatus changed to ON
then
        if(AlarmOnOff.state == ON){
        sendTelegram("OHR8bot", "Movimiento en piso de ABAJO")
        }
End
```

En este caso, "AlarmOnOff" es un ítem de tipo interruptor ("*switch*") virtual, que no está asociado a ningún dispositivo real, y que es el que controla si la alarma está activada o no.

Otro bloque interesante es el de activación de la alarma, dado que además de habilitar las notificaciones cuando alguno de los sensores cambia de estado también se verifica que todos

los contactos de las puertas estén cerrados (“*CLOSED*”), avisando con un mensaje en caso contrario. Esto permite que, tanto si se programa la conexión del sistema de alarma o se hace de forma manual, el usuario sea informado de que hay algún elemento que pueda necesitar su atención. El código de ésta configuración es:

```
rule "Alarma conectada"
when
  Item AlarmOnOff changed to ON
then
        sendTelegram("OHR8bot", "Alarma conectada")
        if(WindowSwitch1_Status.state == OPEN)
        {
        sendTelegram("OHR8bot", "Puerta PRINCIPAL abierta al conectar la alarma")
        }
        else if(WindowSwitch2_Status.state == OPEN){
        sendTelegram("OHR8bot", "Puerta GARAJE abierta al conectar la alarma")
        }
        else if(WindowSwitch3_Status.state == OPEN){
        sendTelegram("OHR8bot", "Balcón SALON abierto al conectar la alarma")
        }
        else if(WindowSwitch4_Status.state == OPEN){
        sendTelegram("OHR8bot", "Balcón COCINA abierto al conectar la alarma")
        }
end
```

Figura 18. Gateway iluminado y sensor de presencia, de Xiaomi.

Cabe mencionar que el Gateway, al llevar altavoz incorporado, es capaz de reproducir sonidos, lo cual se utiliza en el sistema para proporcionar algunas alarmas sonoras al abrirse y cerrarse las puertas de acceso a la casa.

5.4. ILUMINACIÓN

El módulo de gestión de la iluminación es otro de los más desarrollados en el entorno de automatización, con dispositivos distribuidos por toda la vivienda. Las ventajas de tener este sistema domótico son numerosas: es posible gestionar mejor el gasto eléctrico, evitando por ejemplo dejar luminarias encendidas sin querer; se pueden crear escenas de iluminación, donde distintas lámparas adoptan diversas intensidades e incluso colores en función de la situación (modo TV, modo cena, alta luminosidad, etc.); permite encender determinadas luces en respuesta a una acción, como puede ser la apertura de una puerta o la presencia de una persona en el espacio; e incluso se pueden integrar en el sistema de alarmas o avisos, de modo que determinado evento pueda ser notificado también a través de las luces domésticas mediante un parpadeo, o una atenuación.

Los elementos que integran este módulo, a nivel hardware, son en su mayoría del fabricante Philips: tanto la serie de bombillas Hue –que cuenta con un "*Binding*" propio que se describe a continuación-, en sus versiones de casquillos GU10 y E27, como un plafón de techo fabricado en colaboración con Xiaomi, y que se integra en el kit descrito en el apartado anterior.

También hay otros elementos de la serie "*smart home*" de Itead en su gama SonOff, tanto relés WiFi como interruptores de pared táctiles que también incorporan relés WiFi. La personalización de estos dispositivos y su implementación en el conjunto se comentan también en un subapartado de esta sección.

Y por último también tienen funcionalidad asociada algunos dispositivos Xiaomi pertenecientes al kit visto con anterioridad, cuya funcionalidad se describe al cierre de este epígrafe.

5.4.1. HUE BINDING

Esta unión integra el sistema de iluminación Hue de Philips[47] en el entorno de openHAB, de modo que las distintas bombillas puedan ser controladas desde el sistema a través del propio puente de conexión "*Hue bridge*", que es el encargado de trasladar las órdenes recibidas a través de la red local a los dispositivos Hue conectados a él mediante el protocolo ZigBee.

El proceso de instalación es bastante sencillo. Tras instalar el "*Binding*" mediante la interfaz "Paper UI", se conecta el "bridge" a la red local, siendo recomendable asignarle una dirección IP fija en lugar de una obtenida por DHCP a través de la configuración del *router*. Tras este paso inicial ya se puede añadir como una "Cosa" a openHAB a través del archivo "*hue.things*". El código de configuración es:

```
Bridge hue:bridge:serialNumber [ipAdress="X.X.X.X"]
```

El puente requiere que se pulse su botón central para dar acceso al "*Binding*".

A partir de este momento ya es posible controlar los distintos dispositivos vinculados al puente. Para integrarlos en el sistema basta con definir un "Ítem" para cada una de sus propiedades, asignándoles el número que el puente les haya adjudicado al vincularlos. Por ejemplo, una definición de una bombilla de casquillo E27 "*White and Color Ambiance*" está formado por los distintos campos o propiedades:

[47] https://www2.meethue.com/es-es

- **color**: el color RGB que adoptará la bombilla. Se puede asociar a un interruptor (“*Switch*”) para encenderla (“ON”) o apagarla (“OFF”); a un selector de color (“*Colorpicker*”) para elegir el color deseado; o a un control deslizante (“*Slider*”) para fijar la intensidad del brillo.
- **color_temperature**: la temperatura de la luz blanca, lo que cambia automáticamente la bombilla al tono de blanco elegido, asociándose a un control de “*Slider*” también.
- **alert**: la bombilla parpadea a modo de aviso, pudiendo configurarse a tres valores: desactivado (“*None*”), alerta corta (“*Alert*”) y alerta larga (“*Long Alert*”).
- **effect**: efecto de color cambiante (“*Color looping*”), como si se tratara de una luz de discoteca. Se asocia a un interruptor que activa o desactiva este modo.

El código completo de la definición en el mapa del sitio para toda la funcionalidad de esta bombilla, agrupado en un apartado propio (“*Frame*”) de la interfaz, es:

```
Frame label="Lámpara suelo salón"{
        Switch      item=       Light3_Toggle    label="ON/OFF"
        Slider      item=       Light3_Dimmer    label="Intensidad (dimmer)"
        Colorpicker item=       Light3_Color     label="Color RGB"
        Slider      item=       Light3_ColorTemp      label="Temperatura
                BLANCO"
        Switch      item=       Light3_Alert     label="Alerta"
                mappings=[NONE="None", SELECT="Alert", LSELECT="Long Alert"]
        Switch      item=       Light3_Effect    label="Luz cambiante"
}
```

Y en la aplicación para dispositivos móviles tiene esta apariencia:

Figura 19. Controles de iluminación para una lámpara de suelo con bombilla Hue E27.

5.4.2. DISPOSITIVOS SONOFF

La marca china Itead[48] tiene una amplia gama de dispositivos destinados a la domótica, siendo los más conocidos su serie de interruptores o relés WiFi "SonOff" basados en el chip ESP8266[49], entre los que se encuentran algunos de los integrados en la instalación domótica tratada en este documento:

- **Basic**, un interruptor WiFi de 90-250 Voltios y 10 Amperios.
- **POW**, similar a un Basic pero hasta 16 A. y con un medidor de consumo eléctrico integrado.
- **DUAL**, que en esencia son dos Basic en un mismo dispositivo (tiene dos canales de entrada y salida), con 16 A. de límite.
- **TH10/16**, que es como un POW pero con un sensor de temperatura o humedad (solo uno de ellos puede funcionar en cada momento) en lugar de el de consumo, y con 10/16 A. de intensidad máxima.
- **Touch y T1**, interruptores táctiles, llaves de la luz encastrables en la pared en las mismas cajas que las tradicionales existentes.

Los dispositivos, con su configuración de fábrica, se pueden controlar mediante la aplicación "*EWeLink*", disponible para iOS y Android. Pero ello implica que una parte de la funcionalidad de la instalación dependa de una nube sobre la que el usuario no posee prácticamente control alguno. Y esto puede resultar relevante en el aspecto de la privacidad, pero también en el de la estabilidad: si la compañía decide discontinuar la solución, o sufre un ataque de algún tipo, la instalación doméstica puede verse afectada. La solución adoptada en este caso es la de reemplazar el firmware[50] original de los dispositivos por otro que se adapte mejor a las necesidades del proyecto.

El firmware elegido para los dispositivos es "*ESPurna*", y está desarrollado por el barcelonés Xose Pérez, alias "Tinkerman"[51]. El proceso de instalación en los dispositivos SonOff (comúnmente conocido como "*flashear*") se ha ido simplificando con el paso del tiempo: inicialmente era necesario abrirlos, soldar pines de conexión y *flashear* a través de un interfaz USB-SerialTTL, pero actualmente es posible hacerlo incluso a través de la conexión WiFi interceptando las actualizaciones OTA ("*Over The Air*") con el servidor de Itead. [6]

La ventaja del nuevo firmware es que hace al dispositivo independiente del servidor matriz, añadiendo además numerosas opciones de configuración y personalización, como por ejemplo el comportamiento de los led de los dispositivos (pudiendo apagarse para que no molesten, o encenderlos en determinadas situaciones), asignarles direcciones IP estáticas en la red local, renombrarlos, etc. Pero la mejora más importante para este proyecto consiste en poder controlarlos a través de MQTT, pudiendo definir con sencillez los parámetros de configuración desde la administración del dispositivo.

---

[48] https://www.itead.cc/

[49] Chip WiFi de bajo costo ,compacto, resistente y sumamente eficiente. Compuesto por un microcontrolador y una pila TCP/IP completa. Web oficial:

https://www.espressif.com/en/products/hardware/esp8266ex/overview

[50] Simplificando, el sistema operativo del dispositivo.

[51] https://tinkerman.cat/

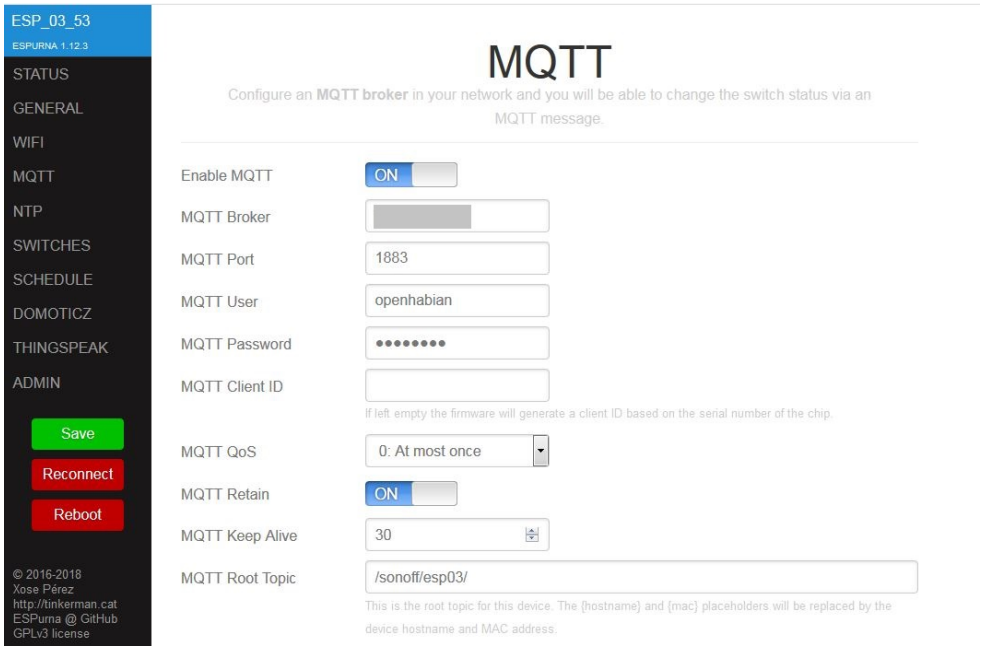

Figura 20. Captura de pantalla de la interfaz de administración del firmware ESPurna, correspondiente a la sección del protocolo MQTT para un SonOff Basic encargado de controlar la calefacción en la vivienda.

De este modo se puede controlar cualquiera de los dispositivos de manera sencilla desde openHAB mediante la publicación de comandos MQTT en los canales pertinentes, y también es posible recibir en el sistema actualizaciones del estado o de los sensores de los dispositivos.

Un ejemplo en uso en la vivienda: una alargadera de cable a la que se ha incorporado un SonOff Basic (llamado "SonOff Test") entre sus dos extremos, de modo que ahora se puede encender y apagar desde openHAB lo que esté conectado a esa alargadera, que normalmente es una lámpara de pie, pero que en la época navideña controla el encendido y apagado del árbol de navidad. El código de control es el siguiente:

```
//Configuración del item en el archive "Sonoff.items"
Switch ProxySonOff_test "Luz pruebas SonOff"
        {mqtt=">[broker:/sonoff/test/relay/0/set:command:ON:ON],
        >[broker:/sonoff/test/relay/0/set:command:OFF:OFF]"}

//Arbol de Navidad con el SonOff Test en el sitemap principal
Switch item=ProxySonOff_test label="SonOff con cable" icon="xmastree_32x32.png"
```

Y ésta es la apariencia que tiene en la interfaz de usuario de la app de dispositivos móviles:

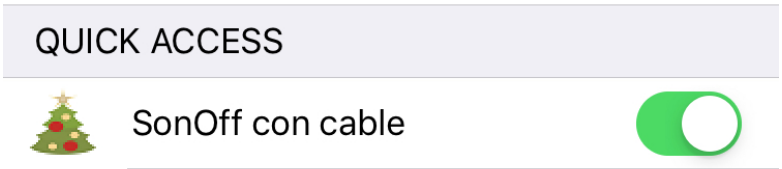

Figura 21. Interfaz de usuario del interruptor WiFi asociado a la iluminación del árbol de Navidad.

También podría programarse una rutina a través de una regla en un archivo "*.rules*" que, por ejemplo, encendiera el árbol todas las tardes a las 18:00 horas y lo apagara a medianoche, de modo que no sea necesario acordarse de encenderlo ni apagarlo cada día.

5.4.3. DISPOSITIVOS XIAOMI

Algunos de los dispositivos pertenecientes al ecosistema de Xiaomi, ya comentado en el módulo de la seguridad, también tienen funcionalidades útiles para gestionar la iluminación doméstica. Estos son los que están integrados en el entorno de producción:

- **Magic Cube Controller**: un "mando a distancia" de forma cúbica que permite definir numerosos gestos, gracias a un acelerómetro alojado en su interior, cada uno de los cuales puede ser asociado a una acción. Por ejemplo, en la instalación actual, los gestos habilitados a través de su definición en el archivo "*xiaomi.rules*" son:
  - Rotación hacia la derecha ("*ROTATE_RIGHT*"). Incrementa un 10% el valor de intensidad de una lámpara, en caso de que esté encendida.
  - Rotación hacia la izquierda ("*ROTATE_LEFT*"). Disminuye un 10% el valor de intensidad de la misma lámpara, si está encendida.
  - Dar la vuelta 90º ("*FLIP90*"). Enciende la lámpara asociada (comando "ON").
  - Dar la vuelta 180º (*FLIP180*"). Apaga la lámpara asociada (comando "OFF").
  - Dos golpecitos ("*TAP_TWICE*"), golpeando dos veces el cubo sobre una superficie rígida. Encender/apagar ("TOGGLE") una segunda lámpara asociada.
  - Agitar ("*SHAKE_AIR*"). Encender todas las luces controladas.
  - Caída libre ("*FREE_FALL*"). Apagar todas las luces controladas.

  Y un ejemplo de código para una de las reglas:

```
//Xiaomi Cube rule
rule "Xiaomi Cube"
when
        Channel 'mihome:sensor_cube:id:action' triggered
then
        var actionName = receivedEvent.getEvent()
        switch(actionName) {
        case "MOVE": {
        }
        case "ROTATE_RIGHT": {
                //Dimmer lámpara +10
                if (Light3_Toggle.state == ON){
                        var dimValR = Light3_Dimmer.state as Number
                        var newValR = dimValR +10
                        if (dimValR <= 90){
                        sendCommand(Light3_Dimmer, newValR)}
                        if (dimValR > 90){
                        sendCommand(Light3_Dimmer, 100)}
                        }
                }
        […]
```

  En la primera parte de la regla se define el canal donde se publica la acción correspondiente cada vez que se mueve el cubo, y se guarda en la variable "*actionName*". Después un bucle "case" lanza la acción correspondiente al tipo de evento recogido, que en el caso del ejemplo es aumentar el "*Dimmer.state*" en 10 unidades, es decir un 10%.

- **Philips Ceiling Lamp**: se trata de un "plafón" de luz inteligente, con conectividad Wi-Fi incorporada y tres temperaturas de luz blanca regulables en intensidad. Además, se

trata de uno de los pocos dispositivos que Xiaomi ha lanzado hasta el momento con capacidad para extender la cobertura de la red ZigBee, haciendo de puente entre elementos de este tipo y el Gateway cuando éste está demasiado alejado de ellos.
Se controla como cualquier otra lámpara regulable: un ítem de tipo deslizante para el brillo, un interruptor para el encendido y apagado, y otro "slider" para el control de la temperatura de color.

- **Sensor de luminosidad** y presencia: ya comentado en el apartado de seguridad anterior, el sensor de luminosidad resulta útil para la implementación del encendido del alumbrado solo en condiciones de oscuridad: para valores del sensor por debajo de un umbral fijado. De este modo es posible encender la lámpara de techo descrita en el punto anterior al detectarse movimiento en la zona a iluminar, durante un intervalo de tiempo configurable (que será como mínimo un minuto, por la limitación del sensor de presencia), y con una intensidad regulable en función de la hora de encendido para, por ejemplo, poder usarse como luz nocturna muy tenue cuando se active de madrugada.
  La definición del ítem en el archivo "*xiaomi.items*" para este tipo de sensores incluirá dos atributos, como en el ejemplo siguiente:

```
//Xiaomi Mi Motion & Light Sensor
Switch MotionLightSensor_MotionStatus <motion> {
        channel="mihome:sensor_motion_aq2:id:motion" }
Number MotionLightSensor_Illumination {
        channel="mihome:sensor_motion_aq2:id:illumination" }
```

Figura 22. Controlador Magic Cube de Xiaomi y lámpara de techo Philips LED, con un sensor de movimiento al fondo.

5.5. CONTROL AMBIENTAL

Este módulo de control ambiental de la vivienda puede considerarse uno de los más importantes de la instalación, dado que gestiona un sistema importante, tanto por el equipamiento involucrado como por su impacto en el confort de los habitantes. La bomba de calor eléctrica, combinada con un panel solar de apoyo para el ACS[52], es el corazón de un sistema totalmente eléctrico.

Figura 23. Bomba de calor y panel solar de apoyo al sistema ACS, en la azotea.

Los equipos desplegados para controlar el entorno también proporcionan información importante a los usuarios sobre la humedad relativa y temperatura en casi todas las estancias de la casa, así como en el exterior de la vivienda.

El control de la calefacción sigue un modelo adaptativo variable en el tiempo para mantener la temperatura objetivo en cada una de las dos plantas, que poseen dos circuitos independientes, en función del momento del día y del día de la semana. La temperatura actual se calcula como una media de las temperaturas registradas en las distintas habitaciones, que a su vez tienen regulado los detentores[53] de las distintas zonas de suelo radiante para que sean lo más similares posibles. Este cálculo de temperatura media ayuda a paliar una de las principales debilidades de los sensores utilizados: su baja tasa de actualización, dado que solo cambian de valor cuando la temperatura media varía más de 0,5ºC.

Se compara esa temperatura media con la objetivo, de forma independiente para cada una de las dos plantas, y si es inferior se habilita el circuito de esa planta a través de uno de los dos interruptores WiFi SonOff Basic instalados en el sistema de bombas. Es posible configurar la

[52] Agua Caliente Sanitaria.

[53] Válvula que regula el caudal de agua en un circuito de calefacción.

frecuencia con la que se realiza esta comparación, que desde hace tiempo se ha dejado marcada a una vez cada dos minutos. Podría ser un intervalo mayor, dado a la importante inercia térmica que presentan los sistemas de calefacción por suelo radiante hace que un tiempo de respuesta más largo no suponga grandes retrasos en los ajustes térmicos, ni un aumento o decrecimiento del gasto apreciables.

A lo largo de los dos últimos años se han probado y analizado distintas estrategias para minimizar el gasto energético manteniendo la temperatura de confort buscada. La información obtenida por la red de sensores y los tiempos de operación registrados han resultado ser extremadamente útiles en la búsqueda de la solución óptima, teniendo en cuenta las particularidades del suelo radiante ya mencionadas: gran inercia térmica y respuesta lenta, de modo que para alcanzar una temperatura objetivo es necesario hacerlo de forma paulatina, y ésta se mantendrá así durante más tiempo con menos intervención del sistema. Se ha comprobado por tanto que resulta mejor mantener una temperatura mínima relativamente elevada incluso cuando no haya ningún usuario en el domicilio, de modo que cuando el sistema tiene que arrancar para alcanzar la temperatura objetivo el tiempo de funcionamiento necesario es más corto, suponiendo un ahorro neto para un mismo periodo.

Esta información también ha ayudado a elegir una tarifa eléctrica sin discriminación horaria, conclusión apoyada también por los informes obtenidos por el medidor de consumo eléctrico que se comentará en el siguiente apartado.

Se está considerando cambiar el actual modelo adaptativo por uno predictivo-adaptativo, de modo que la búsqueda de la temperatura objetivo no esté solo dominada por franjas horarias fijas, sino que además tenga en cuenta otros parámetros. Por ejemplo, considerar también la temperatura exterior permitiría iniciar antes los periodos de caldeo en días muy fríos, o retrasarla en los más calurosos. También se podrían tener en cuenta datos históricos, o incluso la previsión meteorológica.

En la parte de la instalación domótica, los equipos desplegados que tienen la doble finalidad de proporcionar información al usuario y gestionar el funcionamiento de la calefacción, son los siguientes:

- 7 sensores de temperatura y humedad Xiaomi, algunos también con sensor de presión atmosférica incorporado. Están distribuidos en las estancias más utilizadas de la casa.
- 2 interruptores SonOff Basic para controlar la apertura y cierre de los circuitos de calefacción.
- 1 interruptor SonOff TH10, que, además de controlar el encendido y apagado de una lámpara o un ventilador, proporciona información sobre la temperatura ambiente en esa estancia.

También se pueden incluir en este ámbito dos actuadores más, encargados de encender y apagar dos ventiladores responsables de la renovación forzada de aire: uno para la vivienda, que genera el intercambio mediante pequeñas aberturas en la carpintería de puertas y ventanas exteriores que permiten el paso de aire de la calle, y varios puntos de extracción; y otro para la ventilación del forjado sanitario del sótano.

Figura 24. Detalles de la salida de aire forzado en un baño (izquierda), y la salida del aire del sótano.

En conjunto, por tanto, se trata de una red de 8 sensores y 5 actuadores distribuidos por toda la casa.

A continuación puede verse, a modo de ejemplo, parte del código empleado para el control de la calefacción, localizado en el archivo "*heating.rules*", y que opera sobre ítems definidos en el archivo "*heating.items*". En él se puede observar una tarea programada ("cron"[54]) que se ejecuta cada dos minutos, y las distintas comparaciones y cálculos efectuados en función de la temperatura media de cada una de las plantas y su correspondiente temperatura objetivo. La comunicación con los actuadores SonOff se efectúa mediante el protocolo MQTT ya visto.

```
//ACTUADOR
rule "Calculo temperatura cada 2 minutos en planta baja y manejo caldera"
when
        Time cron "0 0/2 * * * ?"  // every two minutes
then
        var tempMedia_pB = ((HT_Temperature1.state as DecimalType) +
        (HT_Temperature3.state as DecimalType) + (HT_Temperature5.state as
        DecimalType))/3
        var tempMedia_pP = ((HT_Temperature2.state as DecimalType) +
        (HT_Temperature4.state as DecimalType))/2

        var tempObjetivo_pB = GF_Heating_TargetTemp.state as DecimalType
        var tempObjetivo_pP = FF_Heating_TargetTemp.state as DecimalType

        logInfo("heating.rules", "LOG: Temperatura media en planta baja es " +
        tempMedia_pB.toString + tempObjetivo_pB)

        if (tempMedia_pB < tempObjetivo_pB){
                sendCommand(ProxySonOff_esp01, ON)
        }
        if (tempMedia_pB >= tempObjetivo_pB){
                sendCommand(ProxySonOff_esp01, OFF)
        }

        logInfo("heating.rules", "LOG: Temperatura media en planta primera es " +
        tempMedia_pP.toString + tempObjetivo_pP)
        if (tempMedia_pP < tempObjetivo_pP){
                sendCommand(ProxySonOff_esp02, ON)
```

[54] Gestor de tareas programadas en Linux.

```
        }
        if (tempMedia_pP >= tempObjetivo_pP){
                sendCommand(ProxySonOff_esp02, OFF)
        }
end
```

En el caso de la activación de la renovación del aire del sótano, el código encargado de gestionarlo se encuentra en el archivo "*timers.rules*", y está compuesto por dos reglas:

```
rule "Enciende Ventilador Sótano"
when
        Time cron "0 0 2 ? * *" //Cuando sean las dos de la mañana
then
        BoundSonOff_esp03.sendCommand(ON)
        logInfo("SonOff.rules", "LOG: Proxy_esp03 ON by schedule")
        end

rule "Apaga Ventilador Sótano"
when
        Time cron "0 0 6 ? * *" //Cuando sean las seis de la mañana
then
        BoundSonOff_esp03.sendCommand(OFF)
        logInfo("SonOff.rules", "LOG: Proxy_esp03 OFF by schedule")
end
```

Es decir, el ventilador del sótano se enciende todas las noches a las 2 de la mañana, y se apaga a las 6 tras cuatro horas de funcionamiento (mostrando un mensaje informativo en los *logs*), que se ha comprobado son suficientes para eliminar la condensación que se acumulaba antes. De esta manera se ahorra electricidad no teniendo el ventilador (50W.) funcionando las 24 horas, y además se elimina el molesto ruido que genera en el jardín durante el día, mientras que por las noches resulta inaudible desde los dormitorios. Y siempre con la posibilidad de cambiar esta programación variando dos líneas de código (se podría hacer una interfaz para que cualquier usuario pudiera ajustarlo de forma visual), o incluso activarlo/apagarlo manualmente en cualquier momento.

5.6. GESTIÓN ENERGÉTICA

El último módulo funcional descrito en este trabajo supone una mezcla entre éxito y fracaso para el sistema global. Y precisamente por no haber alcanzado todas las expectativas también resulta conveniente introducirlo en un trabajo de este tipo, dado que de los errores aprenden lecciones iguales o mejores que de los aciertos.

La gestión energética de la instalación eléctrica se consideró importante desde el comienzo del diseño del sistema domótico para la vivienda, máxime cuando se trata del único suministro energético contratado en el hogar. Por ello se instaló un medidor de consumo contrastado: un "*Mirubox V2"* de la marca Mirubee, alojado en la caja de contadores y con tres pinzas amperimétricas "abrazando" el cable de fase de tres de los circuitos principales de la casa.

Figura 25. Imagen del dispositivo Mirubox V2, con una de las tres pinzas amperimétricas instaladas.

El objetivo buscado y alcanzado era integrarlo en el entorno de openHAB, aprovechando la existencia de una API para la consulta de los valores de consumo instantáneos. Con estos registros se podría, además de guardar para su consulta y representación, gestionar diversas operaciones: lanzar un aviso a los usuarios cuando la potencia superara un valor límite, de modo que se pudiera apagar algo antes de un corte de suministro por superar el límite contratado; impedir que alguno de los dispositivos integrados se active cuando su potencia incrementara el consumo actual por encima del límite (por ejemplo, la calefacción), enviando también un aviso; informar de consumos en espera anómalos que podrían indicar la existencia de aparatos olvidados conectados o con problemas; etc.

Pero la mencionada API presenta una limitación impuesta por el fabricante: 1000 consultas al día. Esto hace que la aplicación de algunas de las ideas anteriormente propuestas sea inviable, dado que requieren un periodo de muestreo del orden de un par de segundos, nunca más de 10, lo cual superaría el límite diario incluso si se eliminaran grandes periodos del día de este escrutinio.

Por lo tanto, el uso de este sistema ha quedado relegado a las consultas realizables desde su propia aplicación para dispositivos móviles, en la que las lecturas de consumo son cada dos segundos, lo cual supone una gran ayuda en momentos de exigencia de suministro eléctrico, pudiendo prever cortes por sobrepasar el límite contratado y actuar en consonancia postergando alguna tarea.

Además, proporciona informes detallados de consumo que han ayudado a perfeccionar los algoritmos de control de temperatura, como se ha mencionado anteriormente, estando dedicado un canal del medidor en exclusiva para el sistema de la bomba de calor.

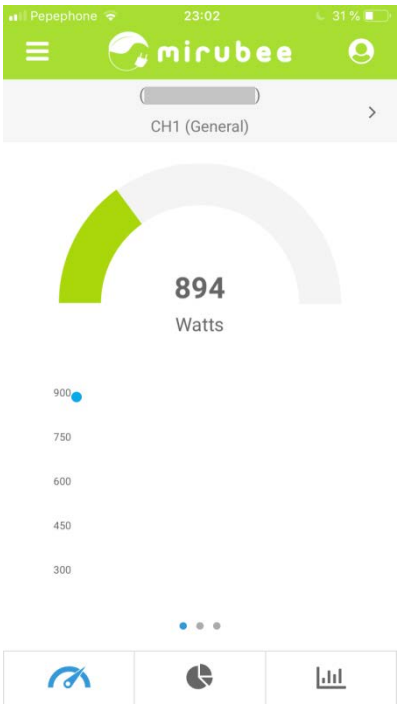

Figura 26. Interfaz del medidor instantáneo de consumo a través de la aplicación de Mirubee.

Figura 27. Informe de la aplicación para un día completo, y para cada uno de los 3 canales.

El medidor incorpora también funcionalidad para controlar instalaciones fotovoltaicas, que es una posibilidad tangible para la vivienda dado que dispone de suficiente espacio en la azotea para la instalación de un buen número de paneles con una orientación adecuada. A tenor de la evolución de los precios de los componentes de una instalación solar para autoconsumo, en continuo descenso y con mejoras importantes de rendimiento, es posible que a medio plazo resulte interesante, y no solo a nivel medioambiental (que ya lo es).

5.7. POSIBLES FUNCIONALIDADES A IMPLANTAR

5.7.1. AUTOMATIZACIÓN DE PERSIANAS

La mecanización mediante motores de las persianas es considerada habitualmente como una de las instalaciones domóticas que más incrementan la satisfacción de los usuarios, convirtiendo la, en ocasiones pesada, tarea de subir y bajarlas a diario en un simple comando en la aplicación de openHAB, que incluso puede ser programada para ejecutarse automáticamente a una hora dada. También permite la apertura y cierre de forma remota para iluminar o ventilar la vivienda incluso sin estar en ella, y puede evitar dejar alguna abierta de manera no deseada mediante un sistema de alertas.

Además de la comodidad también contribuye a la mejora energética de la construcción puesto que se puede combinar con sensores de temperatura o con la previsión meteorológica, dejando pasar al sol para que caliente las estancias en los días más fríos – lo que supone un ahorro en calefacción-, o bajándolas para que no caliente el interior de la casa en los días más calurosos.

Combinando un **motor eléctrico tubular** de persiana con una llave de luz doble ("*two gang*") **SonOff T1** es posible dar una solución de bajo coste a la automatización, sin necesidad de añadir una centralita o módulos adicionales. El firmware ESPurna para este tipo de llaves[55] permite una configuración especial que garantiza que los dos relés nunca estarán activos: si uno se habilita y el otro ya lo estaba, primero lo apaga antes de encender el segundo. Esto asegura que el motor no recibirá simultáneamente las órdenes "subir" y "bajar", que causaría un daño irreparable.

Figura 28. Motor tubular de persiana e interruptores de pared de 1, 2 y 3 canales.

El único inconveniente de la instalación es que para poder alimentar el motor e instalar las llaves en una ubicación lógica será necesario practicar rozas en las paredes para canalizar los cables, que después habrá que cubrir nuevamente, lucir y pintar.

Una alternativa sería la de no instalar llaves, lo que simplificaría la instalación al ser necesario llevar electricidad solo a la caja de la persiana, donde se conectaría a dos SonOff Basic encargados de la lógica de subir y bajar la persiana. Pero esta opción se ha desechado porque por un lado limita la accesibilidad, al ser condición sine qua non disponer de un dispositivo

[55] https://github.com/xoseperez/espurna/wiki/Hardware-Itead-Sonoff-T1

conectado a openHAB, sin posibilidad de manejo físico; una caída de la red WiFi también impediría la apertura; y no se dispondría de la lógica integrada que excluye el envío simultáneo de las ordenes contrapuestas.

5.7.2. GESTIÓN DEL RIEGO DEL JARDÍN Y HUERTO URBANO

Otra de las extensiones del sistema actualmente en fase de estudio e implantación es la de controlar el sistema de riego de la vivienda, que se divide en dos ramales: uno para un pequeño huerto urbano por goteo, y otro para el riego del césped y plantas del jardín mediante difusores.

El sistema estará integrado por una llave general de paso física, a la que seguirá un filtro de partículas para evitar daños en las electroválvulas. Después se instalará una electroválvula maestra, para poder realizar un corte general del circuito de manera remota en caso de pérdidas o avería. A continuación una "T" desdoblará los dos ramales comentados, y en cada uno de ellos se colocará una electroválvula para abrir y cerrar cada circuito de manera independiente.

El control de las electroválvulas, que normalmente son de 24 V., podría realizarse mediante un módulo "***SonOff Pro 4CH***"[56]: un interruptor de cuatro canales independientes que puede ser alimentado y conmutar señales AC entre 90 y 250 V., o señales DC entre 5 y 24 V., por lo que es adecuado para casi cualquier tipo de electroválvula en el mercado. Solo sería necesario resolver cómo alimentarlo con la misma tensión nominal de las electroválvulas elegidas, por lo que si se optara por modelos de 24 V. será necesario conectar un transformador AC-DC para alimentar al SonOff Pro.

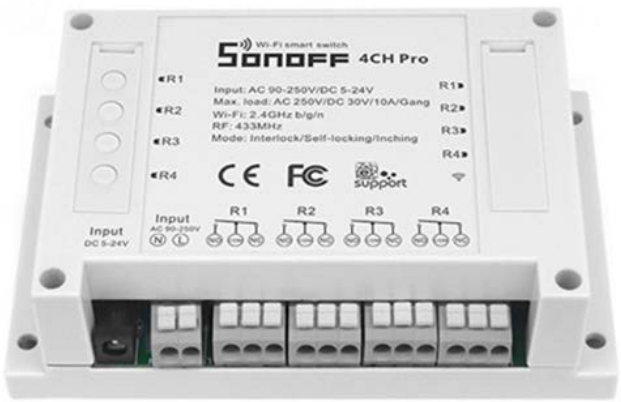

Figura 29. SonOff Pro 4CH (fuente: Itead.cc).

5.7.3. VIDEOVIGILANCIA

Para cerrar este apartado de expansiones futuras de la funcionalidad del sistema domótico se presenta la opción de integrar alguna solución de videovigilancia a través de una o más cámaras IP.

Desde hace años estas cámaras son bastante populares, ofreciendo incluso la opción de acceso a las señales de vídeo en directo a través de aplicaciones para dispositivos móviles que tratan

[56] https://www.itead.cc/sonoff-4ch-pro.html

las señales en nubes corporativas. Esto supone, nuevamente, una brecha en la privacidad de estas imágenes domésticas, puesto que es imposible garantizar quién tiene acceso a las mismas.

La inclusión de este tipo de cámaras en el entorno de openHAB, acompañada de la inhabilitación del tráfico proveniente de las mismas en la configuración del *router*, garantizaría que solo los usuarios conectados al sistema domótico puedan acceder a los flujos de vídeo, eliminando además la necesidad de tener otra aplicación más instalada en los dispositivos. Incluso se podría añadir lógica avanzada utilizando como evento desencadenante de una o más acciones la detección de movimiento por parte de la cámara, como si de un sensor se tratara, pero que además tendría la posibilidad de mandar un fotograma de vídeo adjunto a la alerta.

Existe ya un "*Binding*" desarrollado por la comunidad de usuarios de openHAB que permite realizar esta integración: "***IpCamera***"[57]. Debe instalarse manualmente, no a través de la "Paper UI", y ya han sido probadas de forma satisfactoria un buen número de cámaras IP de distintos fabricantes.

En la instalación actual la incorporación de esta utilidad posiblemente tendría un propósito claro: funcionar como un "vigila bebé" integrado en el sistema domótico, aprovechando las funcionalidades de alguna de las cámaras específicas para este tipo de tareas. Por ejemplo la "*Foscam Fosbaby P1*", totalmente compatible con la Unión mencionada en el párrafo anterior.

Figura 30. Foscam Fosbaby P1, una cámara IP con funcionalidades extendidas de vigila bebés. (Fuente: Foscam.com).

[57] https://community.openhab.org/t/ipcamera-new-ip-camera-binding/42771

6. CONTROL Y VISUALIZACIÓN

La existencia de una red de dispositivos como la descrita en el capitulo anterior resulta imprescindible para el control y recopilación de información de un sistema domótico. Pero también lo es la gestión de todos los datos recogidos y generados, y su presentación al usuario de una forma cómoda y amigable que le permita gestionar el sistema con eficiencia.

En este capítulo se tratarán otras dos capas de la arquitectura implantada: la capa de datos, describiendo las estrategias y utilidades empleadas para su gestión; y la capa de usuario (que se mezcla en parte con la de datos, a la hora de mostrar gráficos informativos), y que resulta indispensable en el diseño e implementación de las distintas interfaces puestas a disposición de los usuarios.

6.1. ALMACENAMIENTO DE DATOS. PERSISTENCIA DE ESTADOS, ANÁLISIS ESTADÍSTICO Y REPRESENTACIÓN GRÁFICA

Como se indicaba en la introducción del capítulo, la correcta gestión, almacenamiento y tratamiento de todos los datos recogidos (lecturas de los sensores) y generados (en forma de eventos de los ítems) resulta vital en un sistema domótico actual. Es necesario, por tanto, establecer una estrategia adecuada que garantice la integridad de esta información.

En el apartado 5.2 se ha comentado la configuración de los módulos del sistema openHAB empleados en la instalación doméstica de este trabajo, y las bases de datos donde se alojan los datos: RRD4J e InfluxDB. La primera destinada a una gestión a corto plazo de la información, y la segunda para una estrategia a largo plazo.

Precisamente la disponibilidad de datos históricos de gran recorrido en una base de datos estable permite realizar análisis comparativos entre periodos de tiempo concretos para extraer conclusiones que permitan mejorar la lógica del sistema, especialmente a través de la generación de gráficos avanzados como los que puede generar la herramienta ***Grafana***.

Grafana es el programa de código abierto más popular y extendido para la analítica y monitorización de series de datos temporales de una manera sencilla y atractiva[58]. Además resulta compatible con numerosos motores de base de datos para alojar los origines de datos, siendo especialmente eficiente la simbiosis con InfluxDB. Cuenta también con una importante comunidad de usuarios que mantienen y mejoran el funcionamiento de manera continua, siendo también un lugar idóneo donde encontrar ayuda en caso de dificultades.

Su instalación resulta bastante sencilla siguiendo las instrucciones disponibles en su página web, que en este caso concreto se ha realizado en la misma Raspberry Pi que aloja la base de datos, un equipo distinto que el que aloja el sistema openHAB. Tras cambiar las credenciales de administrador por defecto en el primer inicio de sesión, a través de la interfaz web ubicada en el puerto 3000 de la máquina, se puede configurar el origen de los datos. En este caso, al ser la misma máquina, se puede indicar la dirección “localhost:8086”, que es el puerto definido para el motor.

[58] https://grafana.com/

A partir de este momento ya es posible comenzar a generar distintos paneles ("*dashboards*") donde se pueden añadir series de datos, configurando los ejes de visualización, el tiempo de refresco y muchas otras opciones.

En las siguientes imágenes se puede ver un ejemplo que recoge la evolución de la temperatura en el salón de la vivienda, una medida de la temperatura exterior, la temperatura objetivo de la planta baja, trazando también el estado de la caldera con los valores del eje Y derecho (ON-1/OFF-0).

Figura 31. Gráfica de la plataforma Grafana para el mes de marzo de 2019.

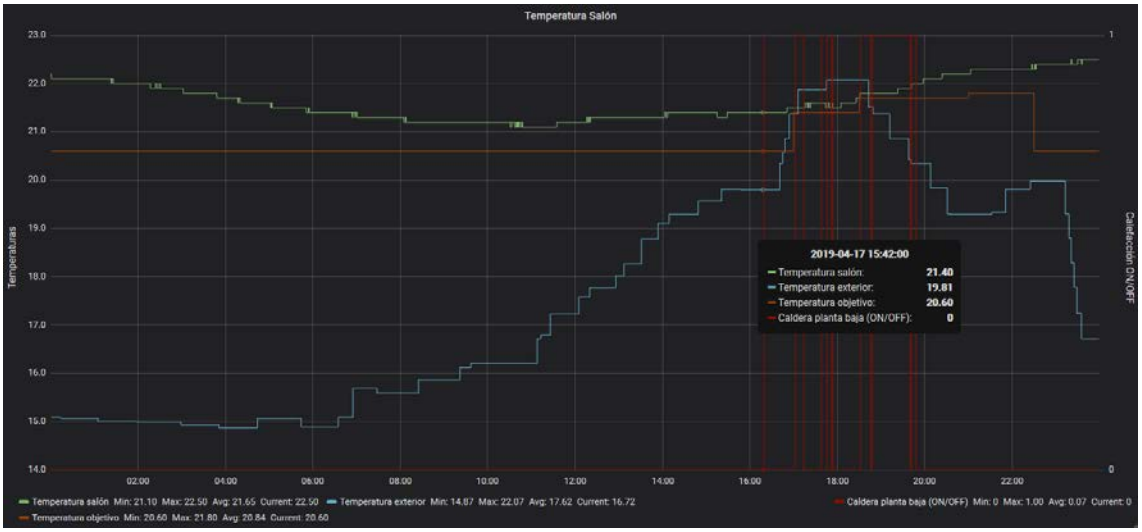

Figura 32. Gráfica para un día del mes de abril de 2019.

Se pueden configurar diversos paneles que se ajusten a las distintas necesidades en cada caso, incluyendo no solo elementos de tipo gráfico como los mostrados, sino otros tipos de representaciones como un dato único, una tabla, mapas de calor, listas, etc., por lo que la complejidad de los cuadro de mandos puede ser tan grande como resulte conveniente.

Los cuadros de mandos o gráficos generados son accesibles a través de cualquier navegador web, y también pueden ser integrados en algunas de las interfaces de usuario disponibles para openHAB, como por ejemplo *HABPanel* o la aplicación para dispositivos móviles.

6.2. APLICACIÓN PARA DISPOSITIVOS MÓVILES (iOS Y ANDROID)

La plataforma openHAB dispone de una aplicación para dispositivos móviles que permite a los usuarios acceder a la información y sistemas de control instalados en el entorno que estén agregados a alguno de los mapas de sitio configurados ("*Sitemaps*").

Esta aplicación está disponible tanto para sistemas Android como iOS. Las capturas de pantalla que ilustran este subapartado se corresponden a un teléfono inteligente y una tableta con el sistema operativo de Apple.

El funcionamiento es sumamente intuitivo. La información se muestra en diversas pantallas, que se configuran a través del diseño de los mapas, y las acciones se ejecutan tocando sobre los distintos controles: interruptores, deslizantes, selectores de color, etc. Se puede cambiar entre pantallas seleccionando el icono de la esquina superior derecha.

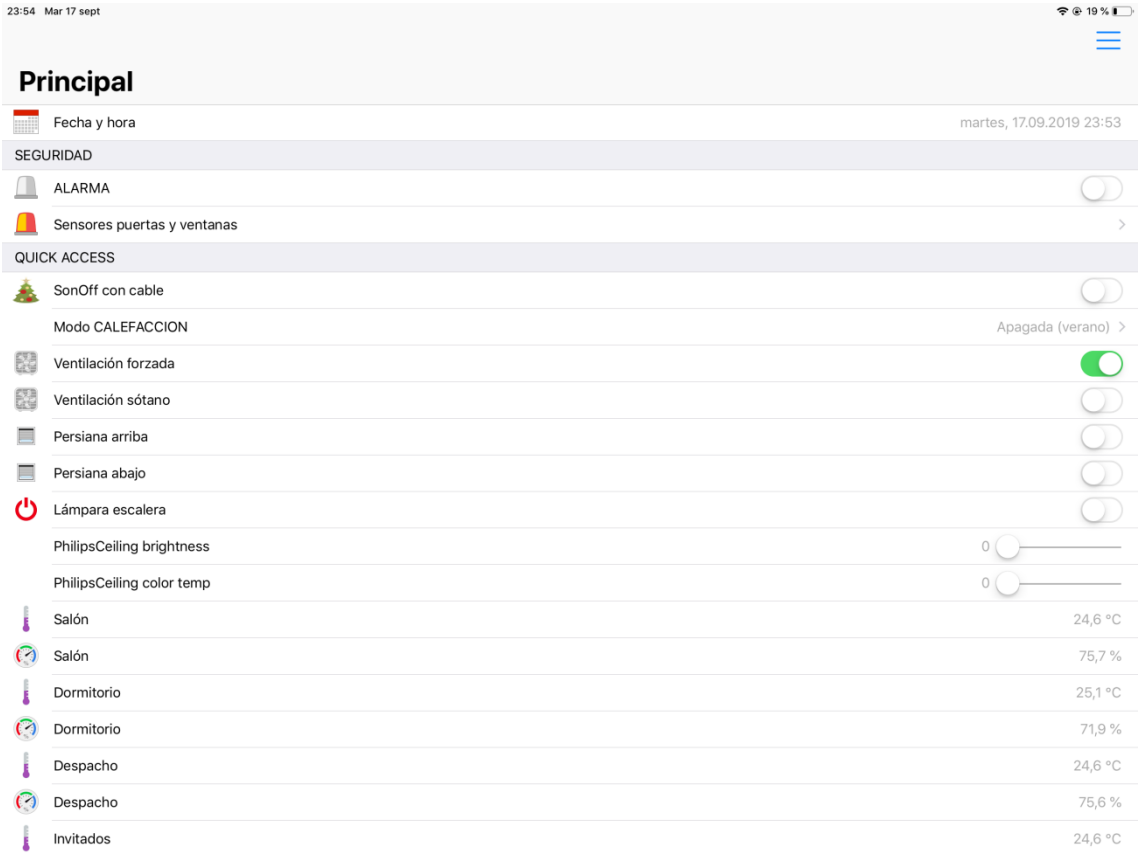

Figura 33. Pantalla principal del sistema mostrado en la aplicación de openHAB para iPad.

La definición de elementos y grupos en los archivos de mapa de sitio es bastante sencilla dado que basta con hacer referencia a los elementos definidos en los archivos de ítems. Por ejemplo, los controles de los ventiladores de la ventilación forzada y del sótano que se pueden ver en la anterior figura, tal y como están en el archivo "*principal.sitemap*". son:

```
Switch item=sonoff_vent label="Ventilación forzada"          icon="fan"
Switch item=ProxySonOff_esp03 label="Ventilación sótano"  icon="fan"
```

Donde "*Switch*" es el tipo de elemento (un interruptor), "*item*" hace referencia al ítem definido, "*label*" es la etiqueta a mostrar en la pantalla de la aplicación, y "*icon*" el icono

asociado, que puede elegirse entre los disponibles del conjunto predeterminado que acompaña a la distribución[59] o uno personalizado añadido por el usuario. Algunos iconos tienen versiones diferentes para el estado activado y desactivado, resultando todavía más visuales para transmitir información al usuario, como puede observarse en los distintos interruptores de la figura.

En la siguiente imagen se puede ver la pantalla de información y control de la calefacción de la vivienda en la aplicación abierta en un iPhone. En este caso es puramente informativa y no permite ninguna acción, por lo que sería necesario cambiar a otro mapa para poder variar el modo de calefacción.

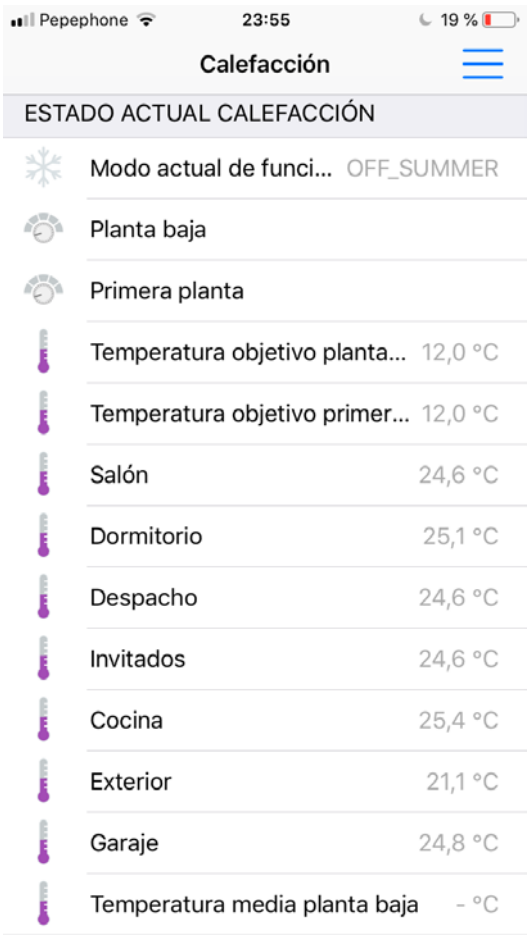

Figura 34. Pantalla informativa del sistema de calefacción en la aplicación de openHAB para iPhone.

El código asociado a los indicadores de estado de cada una de las plantas, recogido en el archivo "*heating.rules*", es:

```
Text item=ProxySonOff_esp01 label="Planta baja" icon="heating"
Text item=ProxySonOff_esp02 label="Primera planta" icon="heating"
```

[59] https://www.openhab.org/docs/configuration/iconsets/classic/

6.3. PANELES DE CONTROL (HABPANEL)

Una de las interfaces más interesantes para gestionar una instalación son los paneles de control suministrados por el módulo "*HabPanel*", que se incluye por defecto en la instalación estándar. Permite generar cuadros de mando amigables, que resultan idóneos para ser utilizados en tabletas u otros dispositivos con pantallas táctiles.

La configuración de los paneles se hace de forma dinámica desde la propia interfaz, sin necesidad de acceder o definir archivos de configuración, y se referencian los mismos ítems usados en el sistema. Pinchando y arrastrando se puede cambiar el tamaño de los elementos y su disposición en la pantalla; además hay numerosos tipos de componentes o "widgets" que pueden ser utilizados, todos ellos con diversos grados de personalización, Algunos de los más habituales y útiles son:

- "**Dummy**", que permite mostrar el estado de cualquier ítem, sin interactividad, acompañado opcionalmente de una descripción y un icono.
- "**Switch**", para controlar interruptores definidos en los ítems. Además de permitir cambiar el estado también informa del mismo.
- "**Label**", para incluir una etiqueta de texto que ayude a maquetar y dar sentido a la información del panel.
- "**Button**", un botón que al ser pulsado desencadenará una acción determinada. Generalmente se usarán varios para presentar diferentes opciones para un mismo ítem.
- "**Slider**", el mismo control deslizante visto con anterioridad para variar el valor numérico de una propiedad asociada a un ítem.

En la documentación oficial[60] se pueden encontrar todos los "*widgets*" permitidos, sus opciones de configuración y algunos ejemplos prácticos de cómo emplearlos en el diseño de cuadros de mando.

Los paneles son accesibles a través de un navegador web, y se puede elegir si la configuración a usar en cada uno es alguna de las almacenadas en el servidor o una personalizada a nivel local. De este modo se pueden tener distintos dispositivos en ubicaciones diferentes con opciones dispares, adecuadas a la funcionalidad que pueda resultar más relevante en su contexto. Por ejemplo, puede ubicarse un "control de iluminación" en una mesa auxiliar del salón, que generalmente será el más susceptible de adoptar distintas escenas lumínicas: modo cena, modo cine, alta luminosidad, etc. Otro panel podría estar ubicado en la primera planta, y estar configurado por defecto para controlar la apertura o cierre de las persianas, el estado de la alarma y la previsión meteorológica, por ejemplo. Las posibilidades son incontables.

Dado que se ejecutan dentro de un navegador tampoco requieren nada especial en el equipo que vaya a usarse, por lo que una tableta antigua, cualquier televisor con capacidad de navegar por internet, un lector de libros electrónicos, una consola de videojuegos, etc. El único requisito es que esté conectado a la red local de la vivienda.

A continuación, se pueden ver algunos de ejemplos de dispositivos con la interfaz "*HabPanel*" configurada, junto con otros ejecutando la "*app*" de openHAB.

[60] https://www.openhab.org/docs/configuration/habpanel.html

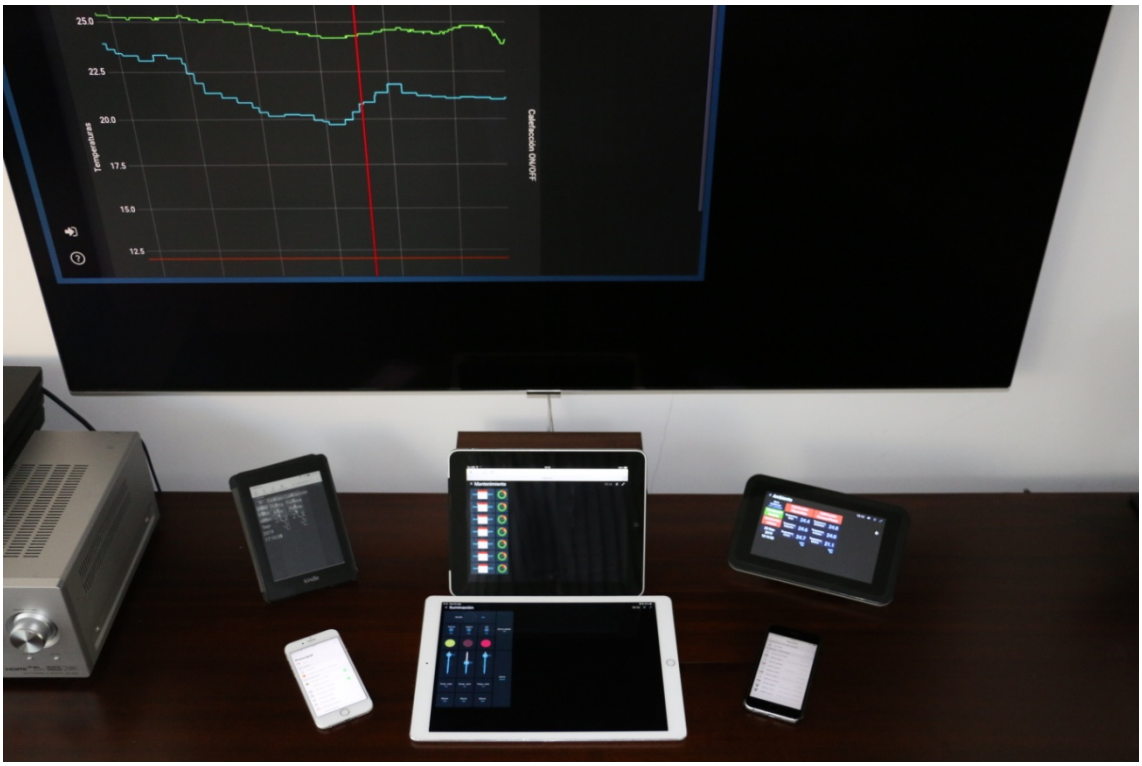

Figura 35. Distintos dispositivos de control empleados en el sistema.

Figura 36. Detalle de los dispositivos de control portátiles: tabletas, teléfonos, lector de libros electrónicos y Raspberry Pi con pantalla.

6.4. INTERFAZ WEB

La última alternativa para el acceso a la información y los controles de openHAB es la de entrar directamente a través de la interfaz web más rudimentaria, conocida como "*BasicUI*", que está vinculada a los mismos mapas de sitio ("*Sitemaps*") que alimentan las aplicaciones móviles comentadas en un subapartado anterior.

Está basada en el lenguaje de diseño "*Material Design Lite*"[61] de Google, que es uno de los estándares de desarrollo de aplicaciones para Android.

Es la más ligera de todas las opciones de control, siendo sus cualidades más destacadas:

- Disposición de elementos responsiva, que permite ajustarse a diferentes tamaños y/o resoluciones de pantalla, por lo que se mantiene usable casi en cualquier dispositivo.
- Navegación AJAX[62], que permite la carga de contenido en una página web de forma dinámica.
- Actualizaciones en directo, por lo que los cambios en los estados de los ítems contenidos en la página se actualizan en tiempo real.

Estas características convierten a esta interfaz en la opción más segura para dispositivos más antiguos o limitados, al prescindir de algunas funciones avanzadas que se han visto en las anteriores.

En la siguiente figura se puede observar el aspecto de la "*BasicUI*" para el mapa de seguridad, "*seguridad.sitemap*".

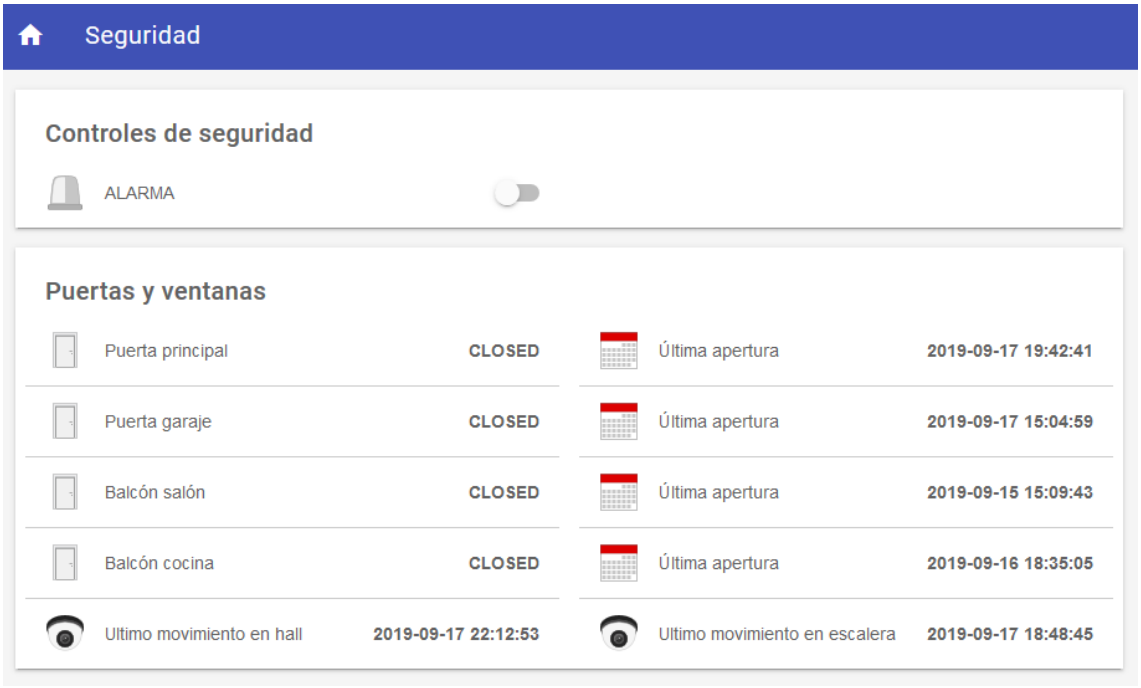

Figura 37. Controles e información de estado de la seguridad de la vivienda, en "BasicUI".

[61] https://getmdl.io/

[62] "Asynchronous JavaScript And XML"

6.5. ACCESO DESDE EL EXTERIOR MEDIANTE VPN[63]

En todo el trabajo se ha realizado especial hincapié en las medidas tomadas para proteger la privacidad de la instalación, así como garantizar la seguridad frente a posibles intrusiones por parte de usuarios no autorizados. Y todo ello, lógicamente, sin que estas precauciones adoptadas impacten negativamente en la usabilidad y comodidad del sistema.

Uno de los mayores inconvenientes reside en la imposibilidad de acceder desde el exterior de la red de área local, cuando no se está conectado a la misma ni por Wi-Fi ni por cable de red, dado que se descarta la opción de hacerlo a través de la "nube" de openHAB por los motivos ya expuestos.

Para sortear esta limitación se ha adoptado una solución personalizada: habilitar un servidor de VPN local basado en la solución de código abierto OpenVPN[64].

Usando una VPN se pueden establecer conexiones de alta seguridad en la comunicación de red, incluso utilizando infraestructura distribuida o pública. [7]

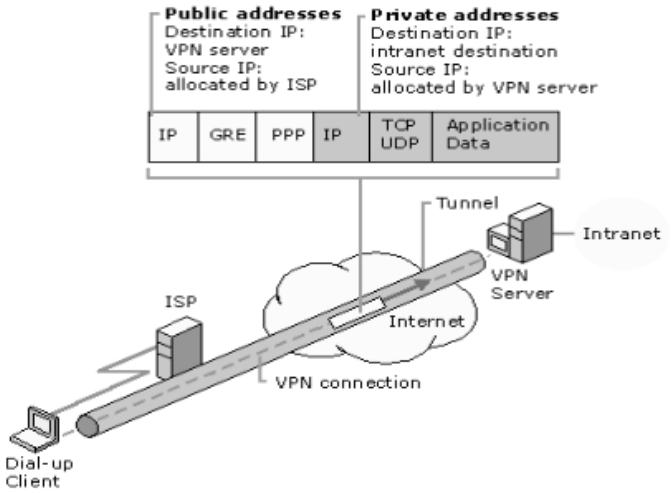

Figura 38. Funcionamiento general de una red VPN. [7]

En resumen: la experiencia de usuario, tras establecer la conexión VPN, es idéntica a la esperada cuando se está dentro de la red local.

La solución de OpenVPN para Raspberry Pi utilizada puede resultar un tanto compleja de instalar y configurar la primera vez, pero que a la larga se ha mostrado muy robusta, con poca necesidad de mantenimiento. También existen diversos tutoriales, e incluso "scripts" que automatizan gran parte del proceso, como es el caso de "*PiVPN*"[65]. Y, nuevamente, una comunidad de usuarios muy activa a la que se puede recurrir para solucionar dudas, aunque lo más probable es que ya hayan sido planteadas y resueltas con anterioridad.

[63] Virtual Private Network, Red Privada Virtual.

[64] https://openvpn.net/

[65] http://www.pivpn.io/

Además de instalar el servidor de VPN en un equipo local (en este caso, la Raspberry Pi de servicios que también aloja la base de datos InlfuxDB, el motor Grafana, y algún otro servicio más), son necesarios algunos pasos adicionales para tener el entorno plenamente funcional.

- Se deben crear credenciales y certificados para cada cliente que quiera conectarse a la VPN. Cada una de ellas tendrá una clave únicamente válida para ese cliente (un ordenador, una tableta, un teléfono, etc.), y llevará asociada una contraseña necesaria para establecer la conexión. Los certificados se crean en el servidor de VPN, se exportan y se importan en la aplicación de OpenVPN de cada dispositivo, que lógicamente habrá tenido que ser instalada previamente.
- Si la conexión doméstica no cuenta con una dirección IP pública estática, sino que es dinámica (que es lo más habitual), será necesario contar con un servicio de DNS dinámico que mapee una dirección DNS conocida con la dirección IP de la casa. De este modo se puede usar ese DNS como referencia para el tráfico de la VPN, encargándose el servicio de actualizar la IP cada vez que el proveedor de internet la modifique. En la instalación se usa el servicio gratuito de NO-IP[66], que solo plantea la limitación de tener un máximo de 3 nombres de sitio y renovaciones mensuales.
- También es necesario que el *router* doméstico tenga abierto el puerto de comunicaciones escogido para "escuchar" el tráfico encaminado al servidor de VPN. Se puede elegir cualquier puerto que no corresponda a ningún otro servicio, siendo indiferente si es TCP o UDP. Cuando el puerto esté abierto será necesario redirigir todo el tráfico que llegue a través de él a la IP interna fija del servidor de VPN.

Con todas estas tareas correctamente realizadas ya es posible establecer la comunicación de cualquier dispositivo con el sistema openHAB desde cualquier ubicación, empleando la aplicación OpenVPN. Se pueden ver ejemplos en las figuras siguientes.

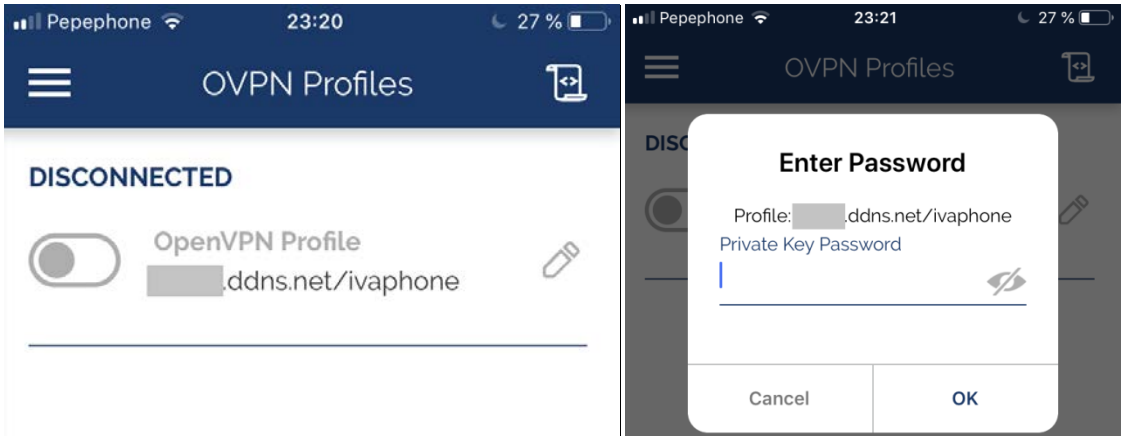

Figura 39. Capturas de pantalla de la aplicación OpenVPN en un iPhone, durante el establecimiento de una conexión con el servidor.

[66] https://www.noip.com/

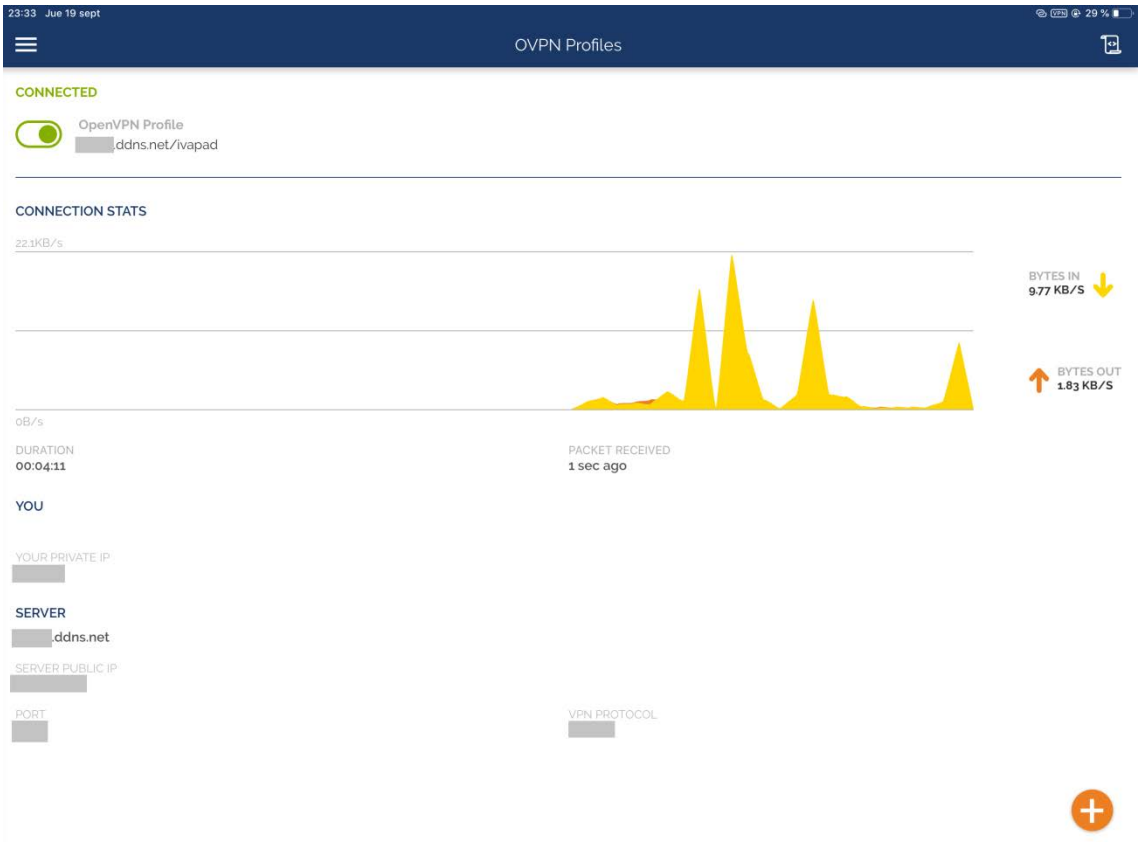

Figura 40. Captura de pantalla de la aplicación OpenVPN en un iPad, mostrando las estadísticas de una conexión ya establecida.

7. CONCLUSIONES

En éste último apartado se intenta hacer una recopilación de todas las lecciones aprendidas durante el proceso de diseño, implantación y pilotaje del proyecto, agrupándolas en 3 grupos: las cosas buenas, que más o menos cumplen con las expectativas generadas; los aspectos negativos asociados, algunos ya previstos, otros inesperados; y aquellos ámbitos en los existe margen para mejorar, siendo uno de los mejores resultados de la realización del presente trabajo.

También se incluye un cálculo aproximado del coste total de la instalación para contextualizar la inversión realizada.

7.1. VENTAJAS

- **Centralización** del control de todo el sistema en un único lugar. Era uno de los objetivos principales buscados, y aunque por el momento no se ha podido conseguir al 100% debido a las limitaciones del medidor de consumo eléctrico instalado, el número de aplicaciones necesarias por los usuarios para el acceso a los dispositivos desplegados se ha reducido drásticamente. Para el día a día, es más que suficiente manejar la "*app*" de openHAB, escoltada por Telegram de forma puntual para las notificaciones y el cliente OpenVPN para establecer una conexión desde fuera.

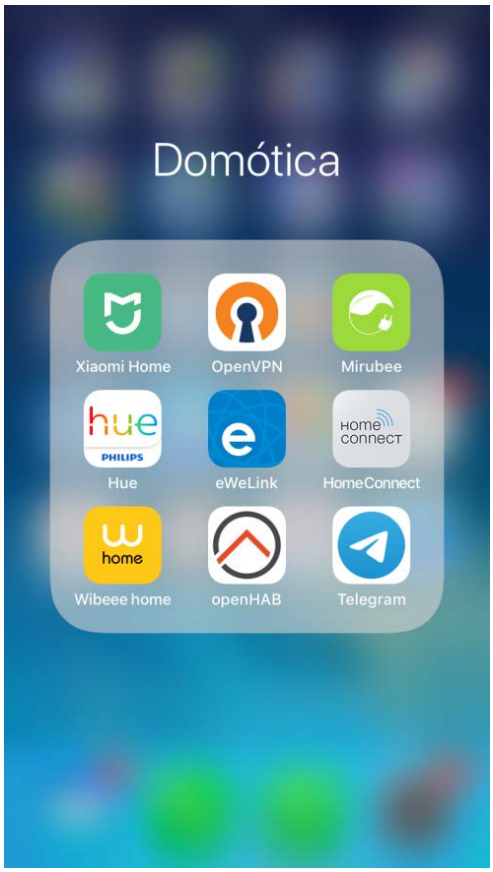

Figura 41. Aplicaciones de domótica instaladas en un iPhone para controlar los distintos sistemas.

- El sistema resulta **sencillo** de utilizar incluso para usuarios inexpertos, y proporciona ciertas **comodidades** que antes no existían, tanto a nivel informativo como operacional.

Por ejemplo, si el ruido de la ventilación resulta molesta en determinado momento ya no es necesario tener que desconectarla desde la ubicación original del interruptor, de difícil acceso. Las tareas programadas también agilizan y simplifican algunas rutinas.

- El **coste** de toda la funcionalidad instalada es bastante **comedido**, sobre todo si se compara con otras soluciones comerciales. Con la ventaja añadida de que se deja de depender de las decisiones estratégicas de los fabricantes, que por ejemplo pueden actualizar un sistema sin previo aviso, cambiando radicalmente la experiencia de usuario o incluso volviéndolo inservible[67].
- Además, se obtiene un control muy elevado sobre la instalación, pudiendo **personalizar** aspectos imposibles de otra manera. Por ejemplo, las llaves de la luz inalámbricas tienen un pequeño led azul que con el firmware original no puede desconectarse, y puede llegar a ser molesto en alguna ocasión; sin embargo, con ESPurna puede programarse para que solo se encienda de día o de noche, esté siempre apagado, parpadee si la llave se desconecta o incluso pase a formar parte del sistema de notificaciones.
- La **privacidad** es mucho más elevada que usando soluciones comerciales, que generalmente almacenan sus datos en alguna nube difícilmente auditable, pueden estar "escuchando siempre", etc. Seguramente alguna de las herramientas usadas también proporcione cierta información a sus desarrolladores, especialmente aquellas instaladas en dispositivos móviles que son más complicadas de controlar, pero la única posibilidad de garantizarla este aspecto al 100% sería renunciar completamente al concepto de casa conectada.
- Otro punto positivo es el **aprendizaje** constante experimentado cuando se decide instalar un sistema así, dado que siempre hay algo que falla, puede mejorarse, necesita una actualización, etc. Podría parecer una debilidad más que una fortaleza, pero para una mente ingenieril apasionada con el mundillo del "hazlo tú mismo" ("DIY"[68]), este desafío difícilmente tendrá connotaciones negativas.

[67] https://www.howtogeek.com/401635/your-expensive-smart-appliance-may-not-last-a-decade/

[68] Do It Yourself.

7.2. INCONVENIENTES

- La **puesta en marcha** resulta muy laboriosa: no es simplemente sacar de la caja, enchufar y listo, como algunas soluciones comerciales. El coste de oportunidad del tiempo necesario puede, en determinadas situaciones, no compensar en absoluto.
- Las tareas de **administración** no están al alcance de cualquiera, y por lo tanto no puede considerarse un sistema para cualquier usuario si no se va a poder contar con un soporte experto para resolver un problema de funcionamiento, reemplazar un dispositivo que falle, actualizar alguna funcionalidad, etc.
- La posibilidad de **replicar** la instalación es **limitada**, dado que los conocimientos necesarios son tan amplios y variados (informáticos, electrónicos, eléctricos, mecánicos, etc.) que podría resultar complicado incluso para perfiles técnicos especializados.
- Siempre existe cierto **riesgo** al poner sistemas delicados de una vivienda, como puede ser la calefacción o una alarma, bajo el control de una solución "DIY", por muy robusta y probada que sea. Aunque los sistemas comerciales tampoco están libres de fallos, así que la diferencia podría no ser tan evidente.
- Con el paso del tiempo podría llegar a ser complejo **reparar** o **sustituir** elementos de la instalación puesto que, al ritmo que avanza la tecnología y con las políticas de renovación constante de los fabricantes, seguramente sea complicado encontrar recambios. Algo de lo que, nuevamente, los productos comerciales tampoco estarán exentos, pudiendo ser incluso peor en algunos casos dado que carecerán de la flexibilidad de las soluciones "DIY".
- Resulta complicado **estar a la última**, salvo con constantes inversiones de tiempo y dinero: siempre habrá algo más bonito, más moderno, con más funcionalidad, etc.

Figura 42. 3 versiones distintas del Gateway Xiaomi, en menos de 5 años.
(Fuente de las imágenes: propia y Gearbest.com)

7.3. POSIBLES MEJORAS

- El **código** de las configuraciones, scripts, personalizaciones, etc. podría ser más **uniforme** y estar mejor explicado. Resultaría más sencillo de entender (¡incluso para el propio autor!), facilitando retomar el desarrollo de los módulos cuando ha pasado cierto tiempo. Esto incluye normalizar más la nomenclatura de los elementos, dado que se arrastran algunas cosas desde las primeras pruebas e instalaciones, y se nota. La documentación suele ser uno de los puntos débiles en los proyectos "DIY", y este no es una excepción.
- Podría emplearse de forma más generalizada la **jerarquía de ítems** en las definiciones de openHAB. Es algo sobre lo que no se ha profundizado, y es posible que simplificarían algunas operaciones y rutinas. Por ejemplo, si todas las luces se encuentran definidas dentro de un mismo grupo resulta más sencilla la orden de "apagar todas", o "comprobar que todas están apagadas", en lugar de verificar su estado una a una.
- Se puede sacar más partido a los **datos** recopilados, a través de análisis gráficos más elaborados para su uso en cuadros de mando. Es posible que de ese modo se pudieran simplificar partes del sistema si se detecta que no se usan como se esperaba, o que no tienen relevancia, etc.
- Los **sistemas de control** tienen un amplio margen de mejora, especialmente enfocado a la experiencia del usuario. Es posible incorporar el control por voz, aunque ya se ha testado con *Siri*[69] en distintos dispositivos con sistema operativo iOS de Apple; finalmente se abandonó por la necesidad física de ese dispositivo, y la brecha en la privacidad del hogar que supone tener habilitado ese tipo de servicios. Pero se podrían explorar alternativas de procesamiento de voz a nivel local en ese ámbito.

Figura 43. Ejemplo de un panel de control más complejo, en una tableta Android.
(Fuente: https://www.domotique-info.fr/2014/09/installation-domotique-frederic-m/)

[69] https://www.apple.com/es/siri/

7.4. COSTE APROXIMADO

El cálculo del coste para el despliegue realizado, contando tan solo el material, resulta complejo de realizar de manera exacta: algunos componentes ya estaban presentes, como es el caso de las Raspberry Pi; otros se han comprado en diferentes momentos, a precios no siempre iguales, como los de SonOff y Xiaomi; y algunos se han usado para diversas pruebas, pero todavía no están integrados en el ecosistema, o puede que nunca lo estén, como el medidor de Mirubee.

En cualquier caso en este apartado se intenta cuantificar este desembolso, de modo que cualquier lector interesado pueda valorarlo como otro elemento a la hora de plantear una instalación similar.

Los precios de compra aproximados de los distintos elementos, sus cantidades y el precio total se recogen en la siguiente tabla.

Artículo	Precio unitario	Nº de unidades	Coste total
Raspberry Pi (incl. Accesorios)	50€	2	100€
Raspberry Pi con pantalla y carcasa	100€	1	100€
Gateway Xiaomi (en kit, aproximado)	30€	1	30€
Sensores Temperatura Xiaomi	11€	7	77€
Sensores magnéticos Xiaomi	8€	4	32€
Sensores presencia Xiaomi	2€	8	16€
Mando a distancia “Botón” Xiaomi	5€	1	5€
Magic Cube Controller Xiaomi	12€	1	12€
Smart Plugs Xiaomi	8€	2	16€
Lámpara techo Philips-Xiaomi	80€	1	80€
SonOff Touch/T1	13€	2	26€
Motores tubulares persiana	30€	2	60€
SonOff Basic	5€	4	20€
SonOff TH10	8€	1	8€
SonOff POW	8€	1	8€
Philips Hue Bridge (en kit, aproximado)	40€	1	40€
Bombilla Hue White and Color E27	20€	3	60€
Bombilla Hue White and Color GU10	20€	3	60€
Bombilla Hue White Ambiance GU10	15€	6	90€
Extensor Wi-Fi TP-Link	15€	1	15€
TOTAL		**52**	**855€**

Figura 44. Precio aproximado de los elementos instalados, y coste total de los mismos.

Es decir, más de 50 elementos en la instalación por un valor inferior a los 900€.

Este coste total de la instalación resulta de considerar solo aquellos dispositivos instalados definitivamente. Incluyendo también los adquiridos para pruebas, desarrollos, accesorios que ya no son necesarios, etc., la cantidad resultante podría aumentar en torno al 10-15%, situándose alrededor de los 1000€ de inversión.

Lógicamente este precio no incluye las horas de trabajo dedicadas: documentación, pruebas, instalación, comparativa de resultados, optimización, etc. Todo desarrollo lleva asociados costes que se producen solo en la primera iteración, pero en este caso concreto resultaría muy complicado cuantificarlos, y prácticamente imposible decidir cómo repercutirlos en instalaciones sucesivas, dado que no se trata de una actividad con fines comerciales.

8. BIBLIOGRAFÍA

[1] L. F. Herrera Quintero, «Smart (Domotic) houses,» *Ingeniería e Investigación, 25(2),* pp. 47-52, 2005.

[2] G. B. a. A. Santokhee, «Power consumption of the Raspberry Pi: A comparative analysis,» de *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, Balaclava, 2016.

[3] mqtt.org, «MQTT.org,» [En línea]. Available: http://mqtt.org/. [Último acceso: 10 09 2019].

[4] ArchWiki, «ArchWiki,» [En línea]. Available: https://wiki.archlinux.org/index.php/InfluxDB_(Espa%C3%B1ol). [Último acceso: 14 septiembre 2019].

[5] J. R.-F. D. Martín Moreno, «ZigBee (IEEE 802.15.4),» 2007.

[6] X. Perez, «Xoseperez / ESPurna,» [En línea]. Available: https://github.com/xoseperez/espurna/wiki/OTA-flashing-of-virgin-Itead-Sonoff-devices. [Último acceso: 2019 Septiembre 16].

[7] A. Skendzic y B. Kovacic, «Open source system OpenVPN in a function of Virtual Private Network,» de *IOP Conference Series: Materials Science and Engineering*, 2017.